GUIDA RAPIDA

MS-DOS® QBasic™

PRINT

DIM

OPEN

PLAY

4.1 8 5.2

KRIS JAMSA

KRIS JAMSA

Titolo originale: *MS-DOS QBasic*

Capo redattore: Massimo Esposti
Redattore: Chiara Tartara

Hanno collaborato a questo volume:

Traduzione: Anna Maria Scarparo, Claudio Nasso
Composizione e impaginazione: L'Angolo Grafico Torre 4 - 20090 San Felice/Segrate (MI)
Copertina: Studio Grafico Margherita, Milano
Stampa: Synthesis Press

ISBN 88-386-0247-6
2ª edizione Febbraio 1993

Printed in Italy
1234567890SPRLLC965432

Indice

Introduzione

Questa guida presenta tutte le istruzioni e le funzioni del QBasic della Microsoft. Ciascuna voce comprende una breve descrizione, la sintassi completa, alcuni particolari sui parametri e, di solito, un frammento del programma che illustra quanto spiegato in precedenza. Inoltre, questa guida contiene informazioni riguardanti le opzioni della riga di comando e una descrizione generale dei vari tipi di QBasic esistenti, delle variabili e degli operatori. La parte che descrive l'utilizzo del QBasic ne introduce l'ambiente e spiega come creare, lanciare e modificare un programma con questo linguaggio.

UTILIZZO DELLA GUIDA RAPIDA

Tutte le funzioni e le istruzioni riportate all'interno di questo volume verranno descritte con il formato riportato nello schema a pagina seguente.

Per la riga di comando verranno utilizzate le seguenti convenzioni:

Convenzione	Descrizione
GRASSETTO	Indica ciò che dovrà venire immesso dall'utente, a meno che non venga racchiuso tra parentesi quadre come spiegato di seguito. Nonostante il QBasic non riconosca la differenza tra maiuscolo e minuscolo, una volta immessi in ambiente QBasic, tutti i comandi vengono visualizzati in maiuscolo.
corsivo	Vengono riportate in corsivo le variabili che devono essere inserite dall'utente volta per volta, come per esempio nomefile o valori numerici.
[*voce*]	Quanto racchiuso tra parentesi quadre può essere inserito facoltativamente dall'utente
{*voce1* | *voce2*}	La parentesi graffa e la barra verticale indicano una scelta che deve essere effettuata tra due o più voci. Dovrà essere scelta una delle due voci a meno che entrambe non siano racchiuse tra parentesi quadre.

voce... — Tre puntini dopo una voce indicano che è possibile aggiungere una o più voci nella stessa forma.

voce
...
voce — Tre puntini tra due istruzioni indicano che è possibile inserire istruzioni aggiunte.

Nome dell'istruzione o funzione

Istruzione BLOAD

Nome dell'istruzione o funzione

Descrizione

Carica in memoria un file di immagine della memoria creato con l'istruzione BSAVE nella posizione specificata.

Sintassi completa

Sintassi

BLOAD *nomefile[,offset]*

Informazioni sui parametri, risultati e utilizzo

Note

- *nomefile* corrisponde a un'espressione stringa che specifica il file contenente l'immagine da caricare.
- *offset* corrisponde all'offset facoltativo dall'inizio del segmento di dati (o ultimo DEF SEG), che indica il punto in cui deve essere caricata l'immagine.
- BLOAD e BSAVE funzionano insieme e forniscono un modo veloce e pratico per caricare valori di array o immagini grafiche.
- Non si utilizzi BLOAD su file creati con BASICA. Il QBasic e BASICA registrano i dati in modo diverso.

Brevi esempi

Esempio

```
'Crea un array che si chiamerà VENDITE
DIM vendite (1 TO 500)

'Imposta l'indirizzo del segmento all'inizio di VENDITE
DEF SEG = VARSEG(vendite(1))

'Carica l'array utilizzando BLOAD
BLOAD "VENDITE.DAT", VARPTR(vendite((1))

'Riporta il segmento di dati al QBasic
DEF SEG
```

Istruzioni e funzioni correlate

Funzioni correlate: VARPTR; VARSEG

Istruzioni correlate: BSAVE; DEF SEG

OPZIONI DELLA RIGA DI COMANDO

Avviando il QBasic dalla riga di comando del DOS, possono essere utilizzate le seguenti opzioni:

Opzione	**Descrizione**
/b	Imposta il video monocromatico.
/editor	Attiva solo l'editor di testo dell'MS-DOS.
nomefile	Carica lo specifico file sorgente Basic (o il file testo se questo è stato utilizzato /editor); aggiunge .BAS (o .TXT per file di testo) al nomefile se viene omessa l'estensione.
/g	Imposta un output a video più veloce.
/h	Imposta la risoluzione massima consentita per la periferica video.
/mbf	Fa sì che i numeri vengano letti e registrati in formato binario Microsoft.
/nohi	Consente l'utilizzo di un monitor che non richieda un supporto ad alta intensità.
/run *nomefile*	Lancia lo specifico file sorgente Basic.

DIFFERENZE TRA IL QBASIC E IL GW-BASIC/BASICA

Caratteristiche linguaggio	QBasic	GW-BASIC/BASICA
Numeri di riga	Facoltativo	Obbligatorio
Istruzione IF	Una riga o un formato blocco	Una riga
Sottoprogrammi	Supportati	Non supportati
Funzioni	Supportate	Non supportate
Record definiti dall'utente	Supportati	Non supportati
Istruzione SELECT CASE	Supportata	Non supportata
Istruzione DO	Supportata	Non supportata
Numeri interi di tipo long (32 bit)	Supportati	Non supportati
Virgola mobile di tipo IEEE	Supportata	Non supportata
Costanti	Supportate	Non supportate
Lunghezza stringa fissa	Supportata	Non supportata
Ricorsione	Supportata	Non supportata
Codice e dati	160 K	64 KB

Caratteristiche di sistema	QBasic	GW-BASIC/A
Modalità video VGA	Supportata	Non supportata
Modalità video Hercules	Supportata	Non supportata
Modalità video Olivetti	Supportata	Non supportata
Interfaccia per cassette	Supportata	Non supportata
Interfaccia mouse	Supportata	Non supportata

Caratteristiche di programmazione	QBasic	GW-BASIC/BASICA
Aiuti Debug	Supportati	Non supportati
Controllo costante della sintassi	Supportato	Non supportato
Finestre	Supportate	Non supportate

Caratteristiche di programmazione	QBasic	GW-BASIC/BASICA
Guida interattiva	Supportata	Non supportata
Interfaccia a menu	Supportata	Non supportata

SOMMARIO DEI TASTI RAPIDI QBASIC

Combinazione tasti rapidi	Funzione
Shift-← ↑ → ↓	Seleziona un carattere.
Shift-Ctrl-← ↑ → ↓	Seleziona una parola
Canc	Cancella il testo selezionato.
Ctrl-G	Cancella un carattere.
Ctrl-Y	Cancella una riga.
Ctrl-Q-Y	Cancella fino alla fine della riga.
Shift-Canc	Taglia il testo selezionato.
Ctrl-Ins	Copia il testo selezionato nella memoria di transito.
Ins	Attiva e disattiva la modalità inserimento.
Home-Ctrl-N	Inserisce una riga sopra quella attiva.
Fine-Invio	Inserisce una riga sotto quella attiva.
Shift-Ins	Incolla il testo dalla memoria di transito.
F4	Visualizza la finestra di output.
Shift-F5	Lancia il programma attivo.
Ctrl-Q-F	Ricerca un testo.
F3	Ripete la ricerca di un testo.
Shift-F1	Richiama la guida di QBasic.
F1	Richiama la guida per una parola selezionata.
F2	Visualizza l'elenco di sottoroutine.

Combinazione tasti rapidi	Funzione
F5	Prosegue l'esecuzione.
F7	Esegue l'operazione indicata fino al punto in cui si trova il cursore.
F8	Esegue una singola istruzione.
F9	Attiva o disattiva il punto di interruzione.
F10	Esegue una singola istruzione o una procedura

TIPI DI QBASIC

Tipo	Descrizione
INTEGER	Ha valore di 2 byte che deve essere compreso tra –32.768 e 32.767.
LONG	Ha valore di 4 byte che deve essere compreso tra –2.147.483.648 a 2.147.483.647.
SINGLE	Valore a 4 byte con 7 cifre significative.
DOUBLE	Valore a 8 byte con 15 cifre significative.
STRING	Corrisponde a una sequenza che può arrivare fino a 32.767 caratteri

TIPI DI VARIABILI

Il nome di una variabile QBasic può contenere fino a 40 caratteri (lettere, numeri e punti). Inoltre è possibile aggiungere uno dei seguenti caratteri per indicare un particolare tipo di variabile:

Carattere	Significato
%	Variabile intera.
&	Variabile intera di tipo LONG.
!	Variabile a precisione singola.

#	Variabile a precisione doppia.
$	Variabile stringa.

I nomi riservati per i comandi, le funzioni o gli operatori Basic non possono essere utilizzati come nomi di variabili. Si ricorda che il QBasic non riconosce la differenza tra formato maiuscolo e minuscolo (per esempio, il nome di variabile *contabilità* e *CONTABILITÀ* risultano identici).

ARRAY

Per creare un array QBasic, si utilizza la seguente sintassi:

```
DIM nomearray ([inizio_indice TO] ultimo_indice[, ...]) AS
nometipo
```

inizio_indice TO *ultimo_indice* corrisponde all'intervallo di valori dell'indice per gli elementi dell'array. Se si omette l'indice di inizio, il QBasic utilizza il valore 0 per default (l'istruzione OPTION BASE permette di impostare l'indice di inizio per default). I tre puntini indicano che il QBasic supporta array multidimensionali. È possibile specificare fino a 60 dimensioni. La seguente istruzione, per esempio, crea un array bidimensionale con 3 righe e 5 colonne:

```
DIM box (1 TO 3, 1 TO 5) AS INTEGER
```

nometipo corrisponde al tipo di array: INTEGER, LONG, SINGLE, DOUBLE o STRING. La dimensione massima degli array è di 64 KB. I valori validi per l'indice variano da –32.768 a 32.767.

COSTANTI SIMBOLICHE

Il QBasic consente ai programmi di riferirsi a costanti simboliche definibili con l'istruzione CONST:

```
CONST size% = 255
```

I nomi delle costanti QBasic seguono le convenzioni per l'assegnazione dei nomi utilizzati per le variabili.

Una volta definita una costante, è possibile utilizzarla all'interno di tutto il programma.

```
DIM a(size%) AS INTEGER
```

Così facendo è possibile semplificare ulteriori modifiche al programma e migliorarne la leggibilità.

NOMI DI ETICHETTE

Per programmi che non utilizzano numeri di righe, il QBasic permette di utilizzare etichette per riferirsi a punti specifici all'interno del programma. Il nome di un'etichetta può contenere fino a 40 caratteri, deve iniziare con una lettera e finire con due punti (:). I nomi riservati per i comandi, le funzioni o gli operatori QBasic non possono essere utilizzati come etichette.

PRECEDENZA TRA GLI OPERATORI

Il QBasic utilizza il seguente ordine di precedenza per gli operatori durante l'esecuzione di operazioni all'interno di una data espressione. Le operazioni sullo stesso livello di precedenza vengono eseguite in ordine da sinistra verso destra.

Ordine di precedenza
Massima
↓
Minima

Aritmetiche
() Espressioni tra parentesi.
^ Elevamento a potenza
– Negazione
*,/ Moltiplicazioni e divisioni
\ Divisioni intere
MOD Modulo (rimanente)
+, – Addizione e sottrazione

Di relazione
=, >, >=, <, <=, <>

Logiche
NOT
AND
OR
XOR
EQV
IMP

SOTTOROUTINE E FUNZIONI

La dimensione massima di una sottoroutine QBasic, in cui è possibile inserire fino a 60 parametri, è di 64 KB. Funzioni e sottoroutine possono essere create utilizzando le istruzioni SUB e FUNCTION.

FILE DI DATI

La dimensione dei file di dati di QBasic viene limitata solo dallo spazio disponibile su disco. È possibile utilizzare i numeri di file da 1 a 255. La dimensione massima del record consentita per un file di accesso casuale è di 32.767 byte. Il numero di record più alto è 2.147.483.647.

ISTRUZIONI MULTIPLE

È possibile inserire istruzioni multiple su una singola riga di comando se le si separa con due punti (:).

Utilizzo del QBasic

Il QBasic è un ottimo ambiente per creare, controllare e lanciare programmi in Basic. Prima di passare a descrivere le capacità avanzate di QBasic, verranno descritte brevemente le modalità con cui creare, lanciare e modificare un file QBasic.

AVVIO DI QBASIC

Il QBasic può essere avviato cliccando con il mouse sulla voce QBASIC.EXE dallo Shell del DOS o digitando il comando QBasic al prompt del DOS e premendo Invio. Se si avvia il QBasic dallo Shell del DOS, si prema semplicemente Invio al prompt per un nomefile QBasic.

Il QBasic visualizza innanzitutto una schermata che richiede di premere Invio se si desidera accedere alla Guida, e un elenco delle operazioni essenziali di QBasic. Si prema Invio per visualizzare l'elenco delle operazioni possibili. Dopo aver letto quanto riportato, si prema Esc per uscire. Il QBasic visualizzerà la schermata di Figura 1.

La barra dei menu, che viene visualizzata in alto sullo schermo, contiene i menu del QBasic.

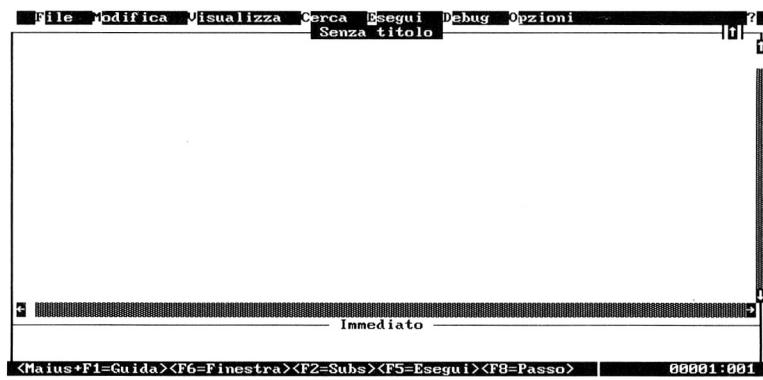



Figura 1 La schermata di apertura di QBasic

- Menu File: consente di salvare o stampare il programma corrente, aprire un programma già esistente per modificarlo o uscire dal QBasic ed entrare nel DOS.
- Menu Modifica: consente di spostare un testo da un punto a un altro di un file o di cancellarne uno che non si intende più utilizzare. Permette inoltre di creare nuovi programmi e funzioni.
- Menu Visualizza: permette di selezionare uno specifico sottoprogramma o funzione, dividere lo schermo in due settori per vedere contemporaneamente due diversi frammenti di un programma o vedere lo schermo di output di QBasic.
- Menu Cerca: consente di ricercare una stringa specificata all'interno di un programma o di sostituirne una con un'altra all'interno di un programma.
- Menu Esegui: consente di lanciare il programma corrente QBasic o proseguire l'esecuzione del programma dopo che questo è stato sospeso a un punto di interruzione di Debug.
- Menu Debug: consente di localizzare gli errori all'interno di un programma consentendo di lanciarlo con un'istruzione alla volta. Permette inoltre di impostare e cancellare i punti di interruzione e di impostare le nuove istruzioni che devono essere eseguite.
- Menu Opzioni: permette di cambiare i colori dello schermo, di definire la posizione del file Guida di QBasic e di attivare o disattivare il controllo automatico della sintassi.
- Menu Guida: consente di accedere ai contenuti, agli indici o alla guida interattiva.

Per aprire un menu QBasic si clicchi sul nome del menu con il mouse oppure si prema il tasto ALT e la lettera corrispondente alla lettera iniziale del nome del menu. Per aprire il menu File, per esempio, si prema Alt-F.

Si apra il menu File. Il QBasic visualizzerà il seguente menu a tendina:

È possibile selezionare un comando all'interno del menu cliccandovi sopra con il mouse, premendo il tasto corrispondente alla lettera evidenziata all'interno del nome di ciascun comando, o evidenziandone il nome con i tasti freccia e quindi premendo Invio. Per cancellare un menu dal video si prema Esc o si clicchi con il mouse all'esterno del menu stesso.

CREAZIONE ED ESECUZIONE DI UN PROGRAMMA QBASIC

Si digiti il seguente programma, premendo Invio dopo ogni riga (a differenza delle versioni precedenti, QBasic non richiede che si digiti il numero della riga prima di immettere l'istruzione. In ogni caso, se si fosse abituati a inserire i numeri di riga o se si disponesse di altri programmi in Basic che li utilizzassero, il QBasic gestirebbe comunque i numeri nel miglior modo). Ci si accerti di aver digitato correttamente ciascuna istruzione e si racchiuda il messaggio della seconda istruzione tra virgolette:

```
CLS
PRINT "Ciao a tutti!"
END
```

L'istruzione CLS pulisce lo schermo, PRINT visualizza il messaggio e END termina il programma.

Si prema Shift-F5 per lanciare il programma. Il QBasic visualizzerà una schermata con l'output di programma e un messaggio *Premere un tasto per continuare*.

Il QBasic suddivide lo schermo in due sezioni: la sezione di ambiente e la sezione di output. Mentre si sta lavorando su un programma, viene visualizzata la sezione di ambiente che contiene la finestra di Programma a sua volta contenente le istruzioni, la barra dei menu QBasic, e la finestra Immediato. Quando si lancia un programma, viene visualizzata la sezione di output. Queste due sezioni permettono di visualizzare l'output precedente del programma mentre si lavora sulle istruzioni del programma stesso. Si prema F4 per visualizzare la sezione di output.

SALVATAGGIO SU DISCO DI UN PROGRAMMA QBASIC

Dopo aver creato un programma, lo si salvi su disco. Si apra il menu File e si selezioni la voce Salva con nome: verrà visualizzato un box di dialogo in cui l'utente dovrà digitare il nome del programma. In questo caso si digiti il nome CIAO.BAS.

Figura 2 Il box di dialogo *Salva con nome.*

L'estensione .BAS indica che il file contiene un programma in Basic. Il nome CIAO descrive la funzione del programma (in questo caso visualizzare un messaggio di saluto).

Si prema Invio per salvare il file. Il QBasic salva il programma, cancella il box di dialogo dal video e visualizza il nome del programma sotto la barra dei menu.

CREAZIONE DI UN NUOVO PROGRAMMA

Dopo aver completato il programma e averlo salvato su disco, si potrà avviare un nuovo programma selezionando la voce Nuovo dal menu File. Il QBasic pulirà la finestra Programma e visualizzerà l'etichetta *Senza titolo.* Si digitino le istruzioni per un nuovo programma e le si salvi come precedentemente descritto.

ESECUZIONE DI UN PROGRAMMA GIÀ ESISTENTE

Per eseguire un programma già esistente o caricarlo nella finestra Programma, si selezioni il comando Apri dal menu File. Il QBasic visualizzerà un box di dialogo che elencherà i file Basic nella directory corrente, come mostrato nella Figura 3.

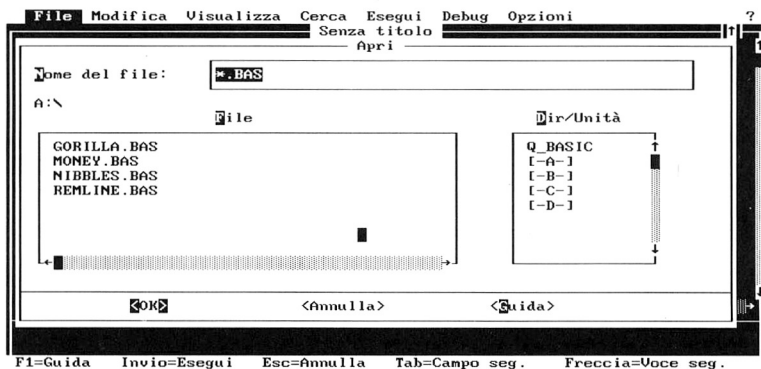



Figura 3 Il box di dialogo *Apri.*

Esistono diversi modi per aprire un programma. Si può semplicemente digitare il nome del programma e premere Invio (se il programma non risiede nella directory attiva, è necessario digitare il percorso completo del file). Oppure, se il nome del file appare nella lista di file, è possibile richiamarlo cliccandovi sopra due volte con il mouse. Ancora, è possibile premere il tasto Tab fino a quando il cursore non verrà posizionato nel box contenente l'elenco di file, evidenziare il nome del file con i tasti freccia e finalmente premere Invio. Se il file desiderato si trovasse in una directory diversa da quella attiva, si prema Tab fino a quando il cursore non verrà posizionato all'interno della finestra Dir/Unità, si selezioni la directory appropriata e si prema Invio. Quando si cambia la directory, il QBasic elenca tutti i file con estensione .BAS presenti in quella directory.

In questo caso si selezioni il programma CIAO.BAS precedentemente creato e il QBasic visualizzerà il contenuto del programma nella finestra programma. Utilizzando i tasti freccia si porti il cursore in vari punti del programma: si noterà che la posizione del cursore verrà segnalata nell'angolo inferiore destro del video.

MODIFICA DI UN PROGRAMMA ESISTENTE

Con il programma CIAO.BAS nella finestra programma, si sposti il cursore alla parola *a tutti*, la si cancelli con il tasto Canc e si digiti il proprio nome. Dopo avere apportato questa modifica, si salvi il programma su disco utilizzando il comando Salva del menu File. A differenza del comando Salva con nome, Salva registra il programma corrente utilizzando il nome esistente.

USCITA DA QBASIC E ACCESSO A MS-DOS

Dopo aver finito di lavorare in QBasic si ritorni al DOS selezionando Esci dal menu File. QBasic registrerà tutte le modifiche apportate al file. Se si tenterà di uscire da QBasic senza aver salvato i programmi creati o le modifiche apportate, o se si proverà ad aprire un altro file senza aver salvato quello precedente, il QBasic visualizzerà un box di dialogo chiedendo se si desidera salvare il programma o no. Se si sceglierà di non salvare le modifiche queste non verranno registrate e l'intero file andrà perduto.

STAMPA DI UN FILE QBASIC

Via via che un programma diventa lungo, può essere utile stamparlo in modo da poterlo leggere più agevolmente e averne una copia su carta. Per eseguire questa operazione, si selezioni la voce Stampa dal menu File e QBasic visualizzerà un box di dialogo in cui verrà richiesto all'utente quale parte del programma desidera stampare. La voce Solo testo selezionato indica a QBasic di stampare solo quelle righe che si sono precedentemente selezionate; la voce Finestra corrente indica a QBasic di stampare solo le istruzioni visualizzate nella finestra di Programma e la voce Programma intero indica a QBasic di stampare tutto il programma.

Il QBasic non indica alla stampante di cominciare ogni nuovo programma su un foglio nuovo e pertanto stamperà sullo stesso foglio fino a quando questo non sarà pieno.

Si selezioni l'opzione appropriata e si prema Invio o si clicchi sull'opzione e quindi su OK se si dispone del mouse.

UTILIZZO DELLA GUIDA INTERATTIVA DI QBASIC

Come già detto, il QBasic contiene una guida interattiva sull'utilizzo dei comandi QBasic. È possibile utilizzare il comando Stampa dal menu File per stampare il contenuto di ogni videata esattamente come si potrebbe stampare un programma QBasic selezionando il testo e avviando il comando Solo testo selezionato dal menu Stampa.

Si prema Shift-F1 per visualizzare la guida interattiva che descrive in quali modi è possibile accedere alla guida QBasic. Le opzioni Sommario e Indice rappresentano le voci principali della guida.

Se si seleziona l'opzione Sommario, verrà visualizzata la schermata di Figura 4.

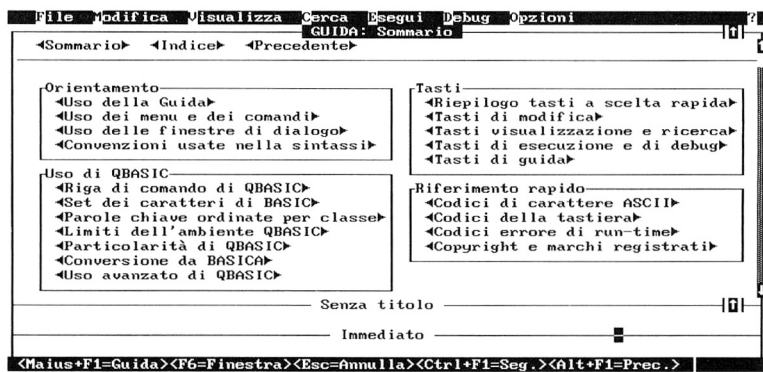



Figura 4 Schermata del contenuto della *Guida* di QBasic.

I comandi e le funzioni vengono raggruppati in quattro gruppi. Il triangolo accanto ad alcune di queste indica che sono disponibili ulteriori informazioni. Per posizionare il cursore alla voce desiderata si utilizzi il tasto Tab e quindi si prema Invio.

Se si seleziona l'opzione Indice, il QBasic visualizza la schermata riportata in Figura 5.

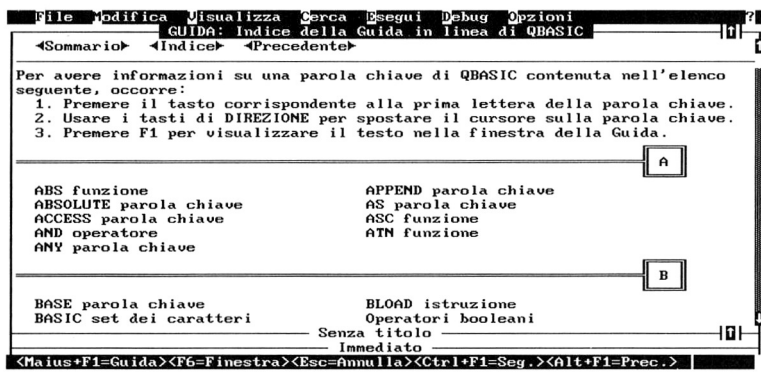



Figura 5 Schermata di indice della *Guida* di QBasic.

Utilizzando i tasti freccia è possibile far scorrere le voci elencate in ordine alfabetico. Se si desiderano informazioni riguardo a una voce che inizi con una data lettera sarà possibile richiamare quella sezione premendo la lettera corrispondente.

Se si utilizza il mouse si potrà far scorrere l'elenco cliccando sulla barra laterale della finestra altrimenti si potrà posizionare il cursore sulla voce desiderata e premere Invio.

La guida QBasic fornisce inoltre un'altra opzione che permette di rivedere velocemente una voce che si è già chiamata. L'opzione Precedente riporta fino a 20 schermate precedenti.

UTILIZZO DELLE FINESTRE DI QBASIC

Il QBasic gestisce tre diverse finestre: la finestra Programma, la finestra Guida e la finestra Immediato.

Le finestre Programma e Guida sono già state descritte: la finestra Immediato, posta sul margine inferiore dello schermo, viene utilizzata per controllare le espressioni mentre si sta scrivendo o verificando un programma QBasic.

La finestra consente di digitare un'istruzione per volta e di visualizzarne immediatamente il risultato. Si supponga di aver bisogno di sapere quante ore ci sono

in un anno: si prema F6 per spostarsi alla finestra Immediato o si clicchi all'interno della finestra con il mouse. Quindi si digiti:

```
PRINT 24*365
```

Alla pressione di Invio, il QBasic visualizzerà immediatamente il risultato (8760). Si prema di nuovo F6 e si ritornerà alla finestra programma.

SCORRIMENTO DI UN LUNGO PROGRAMMA

Via via che i programmi QBasic divengono più complessi, crescono in lunghezza. Il QBasic fornisce diversi modi per spostarsi velocemente all'interno di un programma. Si supponga di dover lavorare con il programma GORILLA.BAS all'interno della directory DOS. Naturalmente sarà possibile utilizzare i tasti freccia per fare scorrere il testo ma l'utilizzo dei comandi all'interno del menu Cerca renderà molto più rapida la ricerca di determinate stringhe. Si apra il menu Cerca e si scelga l'opzione Trova. Il QBasic visualizza un box di dialogo in cui si dovrà digitare la parola o la frase che si desidera localizzare. Sarà possibile selezionare anche la sottoopzione Maiuscolo o Minuscolo o Parola intera se si desidera che venga ricercata una stringa che abbia uno spazio vuoto sia a destra che a sinistra. Per selezionare un'opzione si clicchi su di questa con il mouse o si prema Tab fino a quando il cursore non sarà nella casella desiderata e si attivi o disattivi l'opzione premendo la barra di spazio. Quando l'opzione viene attivata, appare una X tra le parentesi quadre. Per disattivare l'opzione si prema nuovamente la barra di spazio. Se la stringa indicata non dovesse esistere all'interno del programma, verrebbe comunicato all'utente tramite un messaggio. Se l'operazione di ricerca visualizzasse la stringa ma non nel punto in cui si desiderava, la ricerca potrà essere proseguita richiamando l'opzione Ripeti Trova o premendo F3.

VISUALIZZAZIONE DI DUE PARTI DEL FILE

Capita spesso di dover confrontare tra loro due sezioni di un programma. Se un programma è particolarmente lungo, stamparne una copia può essere una soluzione. Altrimenti è possibile aprire una seconda finestra per visualizzare contemporaneamente due frammenti del programma richiamando il menu Visualizza e

scegliendo l'opzione Dividi. La Figura 6 mostra lo schermo diviso in due sezioni, entrambe con una porzione del file GORILLA.BAS.

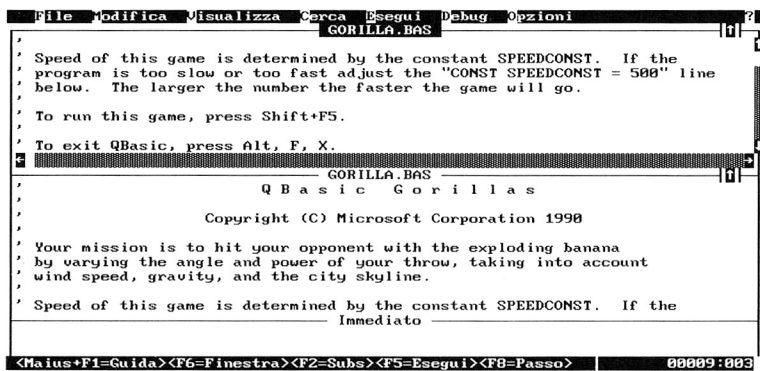



Figura 6 Esempio di visualizzazione contemporanea di due parti del file GORILLA.BAS.

Dopo essersi spostati all'interno di una finestra, si potrà far scorrere il contenuto come precedentemente spiegato. Premendo F6 sarà possibile spostarsi da una finestra a un'altra. Una volta terminato, si selezioni nuovamente il comando Dividi e verrà di nuovo visualizzata una sola finestra.

SOTTOPROGRAMMI E FUNZIONI QBASIC

Generalmente, i programmi lunghi e complessi vengono suddivisi in unità più piccole e maneggevoli, chiamate sottoprogrammi e funzioni. Per ridurre il numero di istruzioni all'interno della finestra programma che bisogna fare scorrere, il QBasic presenta separatamente ciascun sottoprogramma e funzione. Durante la fase di modifica sarà così possibile vedere solo un sottoprogramma o una funzione per volta. Il comando SUBs del menu Visualizza permette di selezionare un sottoprogramma o una funzione specifica per la modifica, per la visualizzazione o per la stampa. Se si seleziona il comando SUBs, il QBasic visualizza il box di dialogo di Figura 7.

Se si dispone del mouse sarà possibile cliccare due volte sul nome del sottoprogramma o della funzione desiderata. Oppure sarà possibile evidenziare il nome

Figura 7 Visualizzazione dei sottoprogrammi del file GORILLA.BAS.

del sottoprogramma o della funzione utilizzando i tasti freccia e premendo Invio. Se si decidesse di non selezionare un sottoprogramma, si prema semplicemente Esc per chiudere il box di dialogo e ritornare alla finestra Programma.

COMANDI TAGLIA, COPIA E INCOLLA

Buona parte delle modifiche che vengono apportate a una programma, vengono eseguite aggiungendo o cancellando istruzioni. In questo caso si potranno utilizzare le opzioni Taglia Copia Incolla del menu Modifica per aggiungere in modo molto più veloce diverse righe.

Si avvii un nuovo programma e si digiti quanto segue,

```
PRINT "Questa è la riga 3"
PRINT "Questa è la riga 4"
PRINT "Cancellare questa riga"
PRINT "Questa è la riga 1"
PRINT "Questa è la riga 2"
PRINT "Anche questa riga deve essere cancellata"
```

Come si può osservare, alcune di queste righe non sono nell'ordine giusto e altre devono essere cancellate. Innanzitutto si selezionino le prime due righe. Se si utilizza la tastiera si posizioni il cursore all'inizio della prima riga, si prema Shift e si prema il tasto ↓ due volte. Se si dispone del mouse, si tenga premuto il pulsante

di sinistra e si trascini il puntatore verso il basso. Ora si apra il menu Modifica e si scelga il comando Taglia per cancellare le righe. Quindi si sposti il cursore sulla riga immediatamente successiva all'istruzione della riga 2, si apra nuovamente il menu Modifica e si avvii il comando Incolla. Il QBasic inserirà le due righe precedentemente tagliate nel punto in cui si troverà il cursore. Ora si cancellino le due righe che non sono più necessarie selezionandole nuovamente e scegliendo l'opzione Taglia dal menu Modifica.

Quando un testo viene tagliato, il QBasic lo rimuove dal punto in cui si trova e lo inserisce nella memoria di transito all'interno del computer. Quando si avvia il comando Copia, il QBasic copia il testo selezionato nella memoria di transito ma non lo elimina dal punto in cui si trovava prima come invece fa il comando Taglia. Quando si avvia il comando Incolla, il QBasic copia il contenuto della memoria di transito nel punto in cui si trova il cursore. Quando si avvia nuovamente il comando Taglia o Copia, il nuovo testo viene sovrapposto al primo nella memoria di transito.

Il comando Cancella elimina il testo selezionato senza copiarlo nella memoria di transito. Dopo aver selezionato Cancella non sarà possibile ripristinare il testo cancellato.

RICERCA E SOSTITUZIONE DI UNA DATA STRINGA

La sostituzione manuale di una stringa su tutto un programma è un'operazione che può richiedere un certo tempo. Il comando Cambia del menu Cerca può accelerare di molto tale operazione. Nel programma che si è appena modificato, per esempio, è possibile utilizzare il comando Cambia per sostituire la stringa *Questa è* con *Output per*.

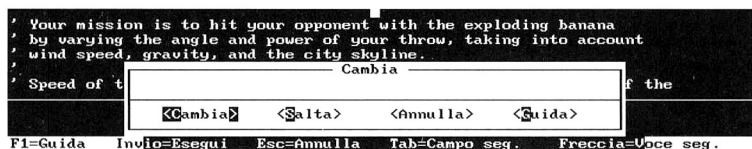
Per iniziare si apra il menu Cerca e si scelga l'opzione Cambia. Il QBasic visualizzerà il box di dialogo di Figura 8.

Al prompt Trova si inserisca *Questa è* e si prema Tab per spostarsi nel campo Cambia in e si digiti *Output per*. Come per il comando Trova è possibile indicare di ricercare solo la stringa in formato maiuscolo o minuscolo o per Parola unica. L'opzione Trova e verifica indica al QBasic di richiedere conferma prima di effettuare la sostituzione mentre l'opzione Cambia effettua le modifiche automaticamente, senza richiedere conferma all'utente. L'opzione Annulla interrompe le operazioni di ricerca e sostituzione. Se si preme Invio dopo aver immesso la stringa nel campo Cambia in viene selezionata l'opzione per default Trova e verifica.

```
File  Modifica  Visualizza  Cerca  Esegui  Debug  Opzioni                 ?
                           GORILLA.BAS
'
' Speed of this game is determined by the constant SPEEDCONST.  If the
' program is too slow or too fast adjust the "CONST SPEEDCONST = 500" line
' below.                      Cambia
'
' To run t  Trova:
'
' To exit
            Cambia in:
'
'
'             [ ] Maiuscole/Minuscole          [ ] Parola intera
'
' Your mis
' by varyi  <Trova e verifica>  <Cambia tutto>  <Annulla>  <Guida>
' wind spe
'
' Speed of this game is determined by the constant SPEEDCONST.  If the
                            Immediato

F1=Guida   Invio=Esegui   Esc=Annulla   Tab=Campo seg.   Freccia=Voce seg.
```

Figura 8 Il box di dialogo presentato dall'avviamento della funzione di ricerca e sostituzione.

Per questo esercizio si selezioni l'opzione per default premendo Invio e il QBasic localizzerà la prima stringa *Questa è* e visualizzerà il seguente messaggio:

Se si seleziona l'opzione Cambia, il QBasic sostituirà il testo; se si seleziona Salta il QBasic passerà alla stringa successiva e se si seleziona Annulla l'operazione terminerà.

VERIFICA DI UN PROGRAMMA QBASIC

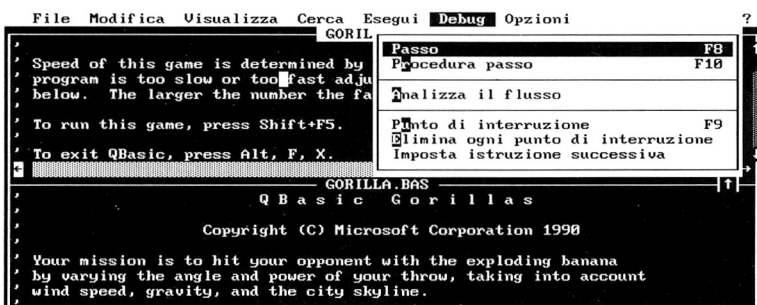
Il debugging è una procedura attraverso la quale si controlla che non ci siano errori in un programma. Il QBasic permette di ricercare due tipi di errori: errori di sintassi ed errori logici. Generalmente, un errore di sintassi è un errore di digitazione, come per esempio l'omissione delle virgolette alla fine di una stringa di caratteri. Quando il QBasic incontra un errore di sintassi, visualizza un box di dialogo con un messaggio di errore.

Per aiutare a ridurre il numero di errori di sintassi in un programma, il QBasic supporta un controllo dei comandi istruzione per istruzione che può essere eventualmente disattivato con il comando Verifica sintassi nel menu Opzioni. Il controllo della sintassi è attivato per default.

Se un programma viene eseguito ma in modo non corretto, ciò significa che contiene almeno un errore logico. Trovare errori logici è molto più difficile che trovare quelli di sintassi. In passato i programmatori erano soliti inserire istruzioni PRINT lungo tutto il programma, in modo da poter trovare eventuali errori logici più facilmente. Il QBasic fornisce un discreto numero di aiuti debugging che permetteranno di trovare errori logici senza ricorrere al comando PRINT.

Per iniziare, si avvii un nuovo programma e si digiti quanto segue:

```
i = 0
increment = 1
WHILE i < 10
      PRINT i
      i = increment
WEND
```

La funzione del programma è quella di visualizzare numeri dallo 0 al 9. Si noterà però che, se si lancia il programma, questo avvia un ciclo continuo che continua a visualizzare il valore 1. Per interrompere il programma si prema Ctrl-Pause. Quindi si apra il menu Debug e il QBasic visualizzerà quanto segue:

Il comando Passo permette di eseguire il programma un'istruzione per volta, dando così la possibilità di osservarne l'output a video o di visualizzare più volte il valore di una variabile nella finestra Immediato.

Il comando Procedura permette di eseguire in una sola volta una serie di comandi che si sa con certezza che sono giusti. Se il programma è stato diviso in sottoprogrammi e funzioni questo comando permette di risparmiare moltissimo tempo.

Il comando Analizza il flusso indica al QBasic di evidenziare ogni istruzione che il programma esegue. Osservando le istruzioni evidenziate è possibile controllare se il programma le esegue nell'ordine desiderato.

Il comando Punto di interruzione consente di attivare o disattivare i punti di interruzione, cioè i punti in cui è possibile interrompere l'esecuzione di un programma. Inserendo punti di interruzione all'interno di un programma (se ne possono inserire quanti se ne vuole) si potrà interrompere un programma immediatamente prima di un punto in cui si sospetta ci sia un errore. Per impostare un punto di interruzione, si porti il cursore sulla riga contenente l'istruzione in cui verrà interrotto il programma e si prema F9. Per cancellare un punto di interruzione si riporti il cursore nella stessa posizione e si prema nuovamente F9.

Il comando Elimina ogni punto di interruzione cancella tutti i punti di interruzione dal programma in una sola operazione.

Imposta istruzione successiva consente di alterare la normale sequenza di esecuzione indicando a QBasic di avviare qualunque comando sulla cui riga si trovi il cursore. Modificando in questo modo il flusso dell'esecuzione sarà possibile superare una sezione di programma che si ritiene sicura e controllarne una successiva.

Inoltre, il comando Esegui contiene tre comandi strettamente collegati al debugging: Avvia, Riavvia e Continua. Il comando Avvia indica al QBasic di lanciare il programma partendo dalla prima istruzione eseguibile. Il comando Riavvia indica al QBasic di cancellare qualsiasi dato sia stato inserito mentre stava eseguendo il programma ed evidenziare il primo comando eseguibile senza però che questo venga avviato.

Il comando Continua permette di proseguire l'esecuzione del programma dall'istruzione corrente. Questo comando è utile quando si riavvia l'esecuzione di un programma dopo averlo sospeso in un punto di interruzione.

Per questo esercizio si avvii il comando Analizza il flusso e quindi si lanci nuovamente il programma premendo Shift-F5. Durante l'esecuzione del programma il QBasic attiverà alternativamente la sezione output e di ambiente, evidenziando ciascuna istruzione che riceverà. In questo caso sarà facile notare che il programma si inceppa nel loop WHILE/WEND. Come in precedenza, si utilizzi Ctrl-Pause per interrompere l'esecuzione. Quindi si utilizzi il comando Passo (o il tasto funzione F8) per eseguire un'istruzione alla volta. Ogni volta che un'istruzione viene completata si prema il tasto F4 per visualizzare la sezione di output o si prema F6 per portarsi alla finestra Immediato, in modo da eseguire l'istruzione PRINT per controllare il valore delle variabili. In breve si scoprirà che il valore della variabile *i* non viene aumentato ma continuamente ripetuto, essendogli stato assegnato il valore *increment.* Infatti, se si modifica l'istruzione *i=increment* con *i=i+increment*, il programma verrà eseguito correttamente.

Il programma di questo esempio è così breve che l'errore avrebbe potuto essere trovato senza aiuti ma, per i programmi più complessi, il debugging del QBasic si rivelerà estremamente utile.

Istruzioni e funzioni QBasic

Funzione ABS

Descrizione

Riporta il valore assoluto di un'espressione numerica.

Sintassi

ABS(*espressione_numerica*)

Note

- *espressione_numerica* corrisponde a una qualsiasi espressione numerica.
- ABS riporta sempre un valore positivo anche se il risultato dell'espressione è negativo.
- ABS riporta un valore dello stesso tipo dell'*espressione_numerica* (cioè di tipo INTEGER, LONG, SINGLE, DOUBLE ecc.).

Esempio

```
PRINT "Valore assoluto di -3*5 è"; ABS(-3*5)
PRINT "Valore assoluto di 3*5 è"; ABS(3*5)
```

Se questo programma viene lanciato si ottiene questo risultato:

```
valore assoluto di -3*5 è 15
```

Funzione ASC

Descrizione

Riporta il valore numerico del codice ASCII per il primo carattere di una stringa di caratteri.

Sintassi

ASC(*stringa_espressione*)

Note

- *stringa_espressione* corrisponde a una qualsiasi stringa di caratteri.

Esempio

```
stringavar$ = "ABC"
PRINT ASC(stringvar$)
PRINT ASC("abc")
```

Se si lancia questo programma si otterrà quanto segue:

```
65
97
```

Funzioni correlate: CHR$

Funzione ATN

Descrizione

Riporta l'arcotangente del valore specificato

Sintassi

ATN(*espressione_numerica*)

Note

- *espressione_numerica* corrisponde al valore per cui si desidera trovare l'arcotangente.
- L'angolo può essere espresso sia in gradi che in radianti. ATN riporta in valore in radianti. Per trasformare i radianti in gradi si utilizzi la seguente equazione:

gradi = *radianti* * (180/3.141593)

Esempio

```
valore = TAN(.7854) 'Tangente di pi/4
PRINT "Arcotangente è"; ATN(valore)
```

Se si lancia il seguente programma, verrà visualizzato quanto segue:

```
Arcotangente è .7854
```

Funzioni correlate: COS; SIN; TAN

Istruzione BEEP

Descrizione

Attiva lo speaker del computer.

Sintassi

BEEP

Note

- BEEP emette un breve segnale sonoro per attirare l'attenzione dell'utente. BEEP non modifica quanto appare a video.

Esempio

```
BEEP
PRINT "Errore nell'accesso al file VENDITE.DAT"
```

Se si lancia questo programma viene visualizzato quanto segue:

(emissione del segnale acustico)

```
Errore nell'accesso al file VENDITE.DAT
```

Funzioni correlate: PLAY

Istruzioni correlate: PLAY, SOUND

Istruzione BLOAD

Descrizione

Carica in memoria un file di immagine della memoria creato con l'istruzione BSAVE nella posizione specificata.

Sintassi

BLOAD *nomefile*[,*offset*]

Note

- *nomefile* corrisponde a un'espressione stringa che specifica il file contenente l'immagine da caricare.
- *offset* corrisponde all'offset facoltativo dall'inizio del segmento di dati (o ultimo DEF SEG), che indica il punto in cui deve essere caricata l'immagine.
- BLOAD e BSAVE funzionano insieme per fornire un modo veloce e pratico per caricare valori di array o immagini grafiche.
- Non si utilizzi BLOAD su file creati con BASICA. Il QBasic e BASICA registrano i dati in modo diverso.

Esempio

```
'Crea un array che si chiamerà VENDITE
DIM vendite (1 TO 500)

'Imposta l'indirizzo del segmento all'inizio di VENDITE
def SEG = VARSEG(vendite(1))

'Carica l'array utilizzando BLOAD
BLOAD "VENDITE.DAT", VARPTR(vendite(1))

'Riporta il segmento di dati al QBasic
DEF SEG
```

Funzioni correlate: VARPTR; VARSEG

Istruzioni correlate: BSAVE; DEF SEG

Istruzione BSAVE

Descrizione

Copia il contenuto di una regione della memoria a un file di output o a una periferica.

Sintassi

BSAVE *nomefile, offset, lunghezza*

Note

- *nomefile* corrisponde a una stringa che specifica il file o la periferica in cui deve essere trasferito il contenuto della memoria.
- *offset* corrisponde al punto all'interno del segmento corrente del primo byte che si desidera copiare.
- *lunghezza* corrisponde al numero di byte che dovranno essere copiati (da 0 a 65.535).
- BSAVE esegue una copia byte per byte. Se si salva il contenuto della memoria in un file, in seguito sarà possibile utilizzare BLOAD per ripristinare l'immagine in memoria.

Esempio

```
'Crea un array con 500 elementi
'inizializza l'array con il valore da 1 a 500
DIM vendite(1 TO 500)
FOR i = TO 500
      vendite(i) = i
NEXT i

'Imposta l'indirizzo di segmento all'inizio delle vendite
DEF SEG = VARSEG(vendite(1))

'Salva l'array utilizzando BSAVE
BSAVE "VENDITE.DAT", VARPTR(vendite(1)), 2000

'Ritorna al segmento dati QBasic
DEF SEG
```

Funzioni correlate: VARPTR; VARSEG

Istruzioni correlate: BLOAD; DEF SEG

Istruzione CALL

Descrizione

Trasferisce il controllo a un sottoprogramma Basic.

Sintassi

CALL *nome_sottoprogramma* [(*lista_argomento*)]
oppure
nome_sottoprogramma[*argomento_lista*]

Note

- *nome_sottoprogramma* corrisponde al nome del sottoprogramma Basic creato utilizzando l'istruzione SUB.
- *lista_argomento* corrisponde a un elenco di parametri separati tra loro da virgole. Il sottoprogramma può modificare il valore dei parametri.
- È possibile avviare un sottoprogramma con o senza la parola chiave CALL. Se si omette CALL non si devono inserire gli argomenti tra parentesi. Inoltre, se si omette CALL sarà necessario utilizzare l'istruzione DECLARE. Se si omettesse una dichiarazione, il QBasic la fornirebbe automaticamente.
- Per evitare che un sottoprogramma modifichi il valore dei parametri, è sufficiente racchiudere il parametro stesso tra parentesi:

  ```
  CALL TEST (a, b, (c))
  ```

- In questo caso il sottoprogramma potrà modificare il valore dei parametri *a* e *b* ma non di *c*.

Esempio

```
DECLARE SUB SwapVal (a,b)
a = 1
b = 2
CALL SwapVal (a, b)
PRINT a; b
END

SUB SwapVal(x, y)
temp = x
x = y
y = temp
END SUB
```

Se si lancia questo programma si otterrà:

```
2 1
```

Istruzioni correlate: CALL ABSOLUTE; CHAIN; DECLARE

Istruzione CALL ABSOLUTE

Descrizione

Trasferisce il controllo a una subroutine in linguaggio macchina.

Sintassi

CALL ABSOLUTE ([*lista_parametri*,]*offset*)

Note

- *lista_parametri* corrisponde a un elenco facoltativo di parametri separati tra loro da virgole.
- *offset* corrisponde alla locazione all'interno del codice segmento corrente all'avvio della procedura.

Esempio

```
'Crea una procedura in linguaggio macchina e
'la avvia utilizzando CALL ABSOLUTE

'Crea un array per registrare il codice macchina
DIM asmroutine(1 TO 6) AS INTEGER

'Dati costituenti la routine del codice macchina
DATI &H55            : ' PUSH BP
DATI &H8B, &HEC      : ' MOV BP, SP
DATI &HB4, 2         : ' MOV AH, 2
DATI &HB2, 65        : ' MOV DL, 65
DATI &HCD, &H21 : ' INT 21H
DATI &H5D            : ' POP BP
DATI &HCB, 0         : ' RET

'Accetta offset di array
offset = VARPTR(asmroutine(1))

'Modifica il segmento all'inizio dell'array
DEF SEG = VARSEG(asmroutine(1))
```

```
'Riempie l'array con il codice macchina
FOR i = 0 TO 11
    READ asmcode
POKE (offset + i), asmcode
NEXT i
'Richiama la routine e ripristina il segmento.
CALL ABSOLUTE(VARPTR(asmroautine(1)))
DEF SEG
```

Istruzioni correlate: CALL

Funzione CDBL

Descrizione

Converte un'espressione numerica in un valore a precisione doppia.

Sintassi

CDBL(*espressione_numerica*)

Note

- *espressione_numerica* corrisponde a un'espressione numerica qualsiasi.
- Utilizzare CDBL è come assegnare un'espressione a una variabile a precisione doppia.
- I valori a precisione singola hanno 7 cifre significative mentre i valori a precisione doppia ne hanno 15.

Esempio

```
PRINT 5 / 6
PRINT CDBL(5 / 6)
```

Se si lancia questo programma viene visualizzato quanto segue:

```
.8333333
.8333333333333334
```

Funzioni correlate: CINT; CLNG; CSNG

Istruzione CHAIN

Descrizione

Trasferisce il controllo da un programma Basic a un altro.

Sintassi

CHAIN *nomefile*

Note

- *nomefile* corrisponde a una stringa di caratteri contenente il nomefile del programma a cui deve essere trasferito il controllo.
- Per scambiare informazioni tra i due programmi concatenati è necessario utilizzare l'istruzione COMMON.
- Dopo aver trasferito il controllo a un altro programma utilizzando CHAIN, il programma chiamato da CALL non riprende il controllo dopo che il programma concatenato è stato eseguito.

Esempio

```
CHAIN "TEST.BAS"
```

Funzioni correlate: CALL; COMMON; RUN

Istruzione CHDIR

Descrizione

Cambia la directory di default per l'unità specificata.

Sintassi

CHDIR *nome_directory*

Note

- *nome_directory* corrisponde alla stringa contenente il nome della directory. Deve essere composta da un massimo di 63 caratteri.
- Per cambiare la directory di default con un'unità diversa da quella corrente, si faccia precedere il nomepercorso dalla lettera dell'unità e due punti (:).

- CHDIR non cambia l'unità di default.
- Non è possibile abbreviare CHDIR

Esempio

```
'Cambia la directory corrente in
'unità C in \DOS
CHDIR "C:\DOS"
```

Istruzione correlata: FILES; MKDIR; RMDIR

Funzione CHR$

Descrizione

Riporta una stringa composta da un carattere che corrisponde al valore ASCII specificato.

Sintassi

CHR$(*valore_ascii*)

Note

- *valore_ascii* corrisponde al codice ASCII del carattere desiderato.
- CHR$ viene utilizzato comunemente per indicare al computer di emettere un segnale acustico. (CHR$(7)).
- I valori ASCII standard vanno da 0 a 127. I PC IBM e IBM compatibili gestiscono inoltre il set di caratteri ASCII esteso da 128 a 255. Molti caratteri estesi sono utili per disegnare box e semplici grafici.

Esempio

```
'Visualizza il set di caratteri ASCII standard ed esteso

FOR i= 0 TO 255
   PRINT i; CHR$(i)
NEXT i
```

Funzioni correlate: ASC

Funzione CINT

Descrizione

Arrotonda un'espressione numerica a un valore intero.

Sintassi

CINT(*espressione_numerica*)

Note

- *espressione_numerica* deve essere compresa tra –32.768 e 32.768. Se l'espressione dovesse risultare al di fuori di questi valori, si verificherebbe un errore di routine.
- CINT arrotonda e non tronca.

Esempio

```
PRINT CINT(34.51)
PRINT CINT(34.49)
```

Se si lancia questo programma verrà visualizzato quanto segue:

```
35
34
```

Funzioni correlate: CDBL; CLNG; CSNG

Istruzione CIRCLE

Descrizione

Disegna un cerchio o un'ellisse con un raggio e un centro specificati.

Sintassi

CIRCLE [**STEP**](*x*, *y*), *raggio* [,[*colore*][,[*inizio_angolo*][,[*fine_angolo*] [,*rapporto*]]]]

Note

- STEP è una parola chiave facoltativa che indica a CIRCLE che i valori di *x* e *y* vengono scostati rispetto alla posizione in cui si trova il cursore.

- *x* e *y* corrispondono alle coordinate del centro del cerchio.
- *raggio* corrisponde al raggio del cerchio nel sistema di coordinate corrente.
- *colore* corrisponde al colore della circonferenza del cerchio. Il cerchio non è colorato all'interno.
- *inizio_angolo* corrisponde all'inizio dell'angolo in radianti per l'arco. Il valore di default è di 2π.
- *rapporto* corrisponde al rapporto tra la lunghezza dell'asse *x* e quella dell'asse *y*. Modificando questo rapporto è possibile creare un'ellisse.

Esempio

```
'Crea dei cerchi in posizioni casuali sul video.
SCREEN 1
FOR i =i TO 100
x = INT(320 * RND)
y = INT(200 * RND)
RADIUS = INT(100 * RND)
CIRCLE (x, y), RADIUS
NEXT i
SCREEN 0
```

Funzioni correlate: POINT

Istruzioni correlate: COLOR; DRAW; LINE; PAINT; PRESET; PSET; SCREEN

Istruzione CLEAR

Descrizione

Inizializza tutte le variabili di programma, chiude i file ed eventualmente definisce le dimensioni degli stack.

Sintassi

CLEAR [, ,*dimensioni_stack*]

Note

- CLEAR chiude tutti i file aperti, imposta le variabili numeriche e gli array a 0 e imposta tutte le stringhe delle variabili a lunghezza zero.

- Se il programma utilizza la ricorsione o esegue diversi livelli di chiamate di subroutine, potrebbe essere necessario aumentare le dimensioni dello stack di programma.
- *dimensioni_stack* corrisponde alla dimensione in byte dello stack. La dimensione dello stack deve essere preceduta da due virgole.
- Lo stack di default è di 2048 byte.
- Non eseguire CLEAR con una subroutine.

Esempio

```
'Inizializza le variabili e crea uno stack di 4096 byte
clear , , 4096
```

Funzioni correlate: FRE

Istruzioni correlate: ERASE

Funzione CLNG

Descrizione

Arrotonda un'espressione numerica a un intero di tipo LONG (4 byte).

Sintassi

CLNG(*espressione_numerica*)

Note

- *espressione_numerica* deve avere un valore compreso tra –2.147.483.648 e 2.147.483.647. Se il risultato è al di fuori di questo intervallo, si verifica un errore di routine.

Esempio

```
PRINT CLNG(338457.8)
PRINT CLNG(2147358.28)
```

Se si lancia questo programma viene visualizzato quanto segue:

```
338458
2147358
```

Funzioni correlate: CDBL; CINT; CSNG

Istruzione CLOSE

Descrizione

Chiude uno o più file o periferiche aperte dall'istruzione OPEN.

Sintassi

CLOSE[[#]*numero_file*[,[#]*numero_file*]]...

Note

- *numero_file* corrisponde al numero assegnato al file o alla periferica nell'Istruzione OPEN.
- È possibile specificare uno o più numeri file in una singola istruzione CLOSE.
- Una volta chiuso un numero di file non sarà più possibile utilizzarlo per operazioni di lettura o di scrittura fino a quando non si aprirà un nuovo file.
- Le istruzioni CLEAR, END, RESET, RUN e SYSTEM chiudono automaticamente il file. Per agevolare la manutenzione, tuttavia, è possibile caricare un'istruzione corrispondente CLOSE per ciascuna istruzione OPEN all'interno del programma.
- CLOSE senza argomenti chiude tutti i file e le periferiche aperti.

Esempio

```
OPEN "TEST.DAT" FOR OUTPUT AS #1
PRINT #1, "Questo è un test"
CLOSE #1
```

Funzioni correlate: OPEN; RESET

Istruzione CLS

Descrizione

Pulisce lo schermo.

Sintassi

CLS[{0|1|2}]

Note

- A seconda della sezione dello schermo che si desidera pulire, CLS presenta quattro opzioni:

Istruzione	Risultato
CLS	Pulisce una finestra contenente grafico o testo.
CLS 0	Pulisce l'intero schermo grafico o di testo.
CLS 1	Pulisce solo la finestra contenente grafico.
CLS 2	Pulisce solo la finestra contenente testo, lasciando la riga di base immutata.

Esempio

```
CLS
```

Funzioni correlate: VIEW; VIEW PRINT; WINDOW

Istruzione COLOR

Descrizione

Imposta il colore dello schermo

Sintassi

COLOR [*principale*][,[*sfondo*][,*contorno*]] (Modalità video 0)
COLOR [*sfondo*][,*tavolozza*] (Modalità video1)
COLOR [*principale*][,*sfondo*] (Modalità video 7-10)
COLOR [*principale*] (Modalità video 4, 12, 13)

Note

- L'istruzione COLOR permette di impostare il colore principale e di sfondo nella modalità testo e i colori della tavolozza in modalità grafica.
- Si veda l'istruzione SCREEN per i valori specifici di ciascuna modalità video.
- Nella modalità video 0, è possibile impostare il colore principale per sedici colori (da 0 a 15). Per utilizzare la versione lampeggiante del colore, si aggiunga il valore 16 al colore ottenendo così un valore da 16 a 31. Il contorno del video di sfondo deve essere un colore il cui valore sia compreso tra 0 e 15.

- Nella modalità 1, è possibile specificare un valore di tavolozza da 0 a 255. La tavolozza specifica quale dei due set di colori verrà utilizzato per visualizzazioni grafiche.
- Nella modalità video da 7 a 10, il colore principale è un numero attributo e il colore di sfondo è un numero di colore.
- Nelle modalità video 4, 12 e 13 il colore principale è un numero attributo. Non è possibile specificare un colore di sfondo.

Esempio

```
SCREEN 0
FOR fcolor = 0 TO 31
    COLOR fcolor
    PRINT "Il colore corrente è"; fcolor
    INPUT vuoto$
NEXT fcolor
```

Funzioni correlate: POINT; SCREEN

Istruzioni correlate: CIRCLE; DRAW; PAINT; PALETTE; PALETTE USING; PRESET; PSET; SCREEN

Istruzione COM

Descrizione

Attiva o disattiva il flusso di dati incanalandoli verso la porta specificata.

Sintassi

COM(n) **ON**
oppure
COM(n) **OFF**
oppure
COM(n) **STOP**

Note

- *n* corrisponde al numero di porte in comunicazione (1 o 2).
- COM ON attiva il passaggio dei dati. Se un carattere arriva alla porta, il programma eseguirà la subroutine definita dall'istruzione ON COM.
- COM OFF disattiva il flusso di comunicazione. I caratteri che arrivano alla porta vengono ignorati.

- COM STOP impedisce il passaggio del flusso di dati fino a quando il programma non esegue un'istruzione COM ON. Le operazioni vengono eseguite una volta che l'incanalamento è avvenuto.

Esempio

```
ON COM(1) GOSUB ComHandler
COM(1) ON
```

Funzioni correlate: ON operazione GOSUB

Istruzione COMMON

Descrizione

Definisce come globali le variabili all'interno di un modulo o tra programmi concatenati.

Sintassi

COMMON [**SHARED**] *lista_variabili*

Note

- La parola chiave SHARED attesta che le variabili vengono divise da tutti i sottoprogrammi e dalle funzioni tramite un modulo.
- *lista_variabili* corrisponde alla lista delle variabili globali con i nomi delle variabili separate da virgole. Il Qbasic permette di specificare variabili come: *nome_variabile*[()][**AS** *tipo*]
- L'istruzione COMMON deve apparire prima di qualsiasi istruzione eseguibile all'interno del programma. Il QBasic associa le variabili in blocchi per posizione e non per nome.

Esempio

```
COMMON a, b, c
a = 1:  b = 2:  c = 3
CHAIN "COMMON.BAS"

'Codice per COMMON.BAS
COMMON x, y, z
PRINT x, y, z
```

Se si lancia questo programma, si ottiene quanto segue:

```
1   2   3
```

Funzioni correlate: STATIC

Istruzione CONST

Descrizione

Definisce una costante simbolica.

Sintassi

CONST *nome_simbolo=espressione* [, *nome_simbolo=espressione*]...

Note

- Le costanti permettono ai programmi di utilizzare nomi simbolici al posto di espressioni numeriche o stringhe.
- *nome_simbolo* corrisponde al nome assegnato alla costante per tutto il corso del programma. Una volta definito, tale valore non potrà essere modificato.
- *espressione* corrisponde a una stringa numerica o di caratteri assegnata alla costante. All'interno di *espressione* non è possibile utilizzare variabili o funzioni.
- Le costanti definite in un sottoprogramma o in una funzione sono valide solo in quel sottoprogramma o funzione.

Esempio

```
CONST vero = 1
CONST giornisett = 7

'Utilizza la costante nella dichiarazione array
DIM giorni(giornisett)
```

Funzione COS

Descrizione

Riporta il coseno dell'angolo specificato.

Sintassi

COS(*angolo*)

Note

- *angolo* è un'espressione numerica che specifica un angolo in radianti.
- È possibile esprimere un angolo in radianti o in gradi. Il programma di trigonometria del QBasic gestisce però solo i radianti. Per convertire i gradi in radianti, si utilizzi la seguente equazione
  radianti = 3.141593 * (*gradi* /180)

Esempio

```
angolo = .785
PRINT "Coseno di"; angolo; "è"; COS(angolo)
```

Se si lancia questo programma viene visualizzato quanto segue:

```
Coseno di .785 è .7073882
```

Funzioni correlate: ATN; SIN; TAN

Funzione CSNG

Descrizione

Converte un'espressione numerica in un valore a precisione singola.

Sintassi

CSNG(*espressione_numerica*)

Note

- *espressione_numerica* corrisponde a un'espressione numerica qualsiasi.
- Utilizzando la funzione CSNG si ottiene lo stesso risultato che si otterrebbe assegnando l'espressione a una variabile a precisione singola.
- Un valore a precisione singola contiene 7 cifre significative.

Esempio

```
a# = 6
b# = 7
PRINT a#/b#, CSGN(a#/b#)
```

Se si lancia questo programma, si ottiene:

```
.8571428571428571    .8571429
```

Funzioni correlate: CDBL; CINT; CLNG

Funzione CSRLIN

Descrizione

Riporta il numero di riga corrente del cursore.

Sintassi

CSRLIN

Note

- CSRLIN riporta il numero di riga del cursore.

Esempio

```
'Registra la riga e la colonna del cursore
salvariga = CSRLIN
salvacol = POS (0)
'Muove il cursore alla riga 10, colonna 20
LOCATE 10,20: PRINT "Messaggio alla 10,20"
'Ripristina il cursore alla posizione precedente
LOCATE salvariga, salvacol
```

Funzioni correlate: POS

Istruzioni correlate: LOCATE

Funzione CVD

Descrizione

Trasforma una stringa di 8 byte (creata con MKD$) in un valore a precisione doppia.

Sintassi

CDV(*stringa_otto_byte*)

Esempio

```
OPEN "STIPENDI.DAT" FOR RANDOM AS#3 LEN = 35
FIELD #3, 27 AS nomi$, 8 AS stipendi$
GET #3, 1
PRINT "DIPENDENTE: "; nome$
PRINT "Stipendi $"; CVD(stipendi$)
CLOSE #3
```

Funzioni correlate: CVDMBF; CVI; CVL; CVS; CVSMBF; MKD$; MKDBMF$; MKI$; MKL$; MKS$; MKSBMF$

Funzione CVDMBF

Descrizione

Trasforma una stringa da 8 byte contenente un valore a precisione doppia (creato con MKDMBF$) da un formato binario Microsoft in formato IEEE.

Sintassi

CVDMBF(*stringa_otto_byte*)

Note

- CVDMBF, CVSMBF, MKDMBF$ e MKSMBF$ sono utili per mantenere i file di dati creati con le versioni precedenti di Basic.

Esempio

Vedi CVD

Funzioni correlate: CVD; CVI; CVS; CVSMBF; MKD$; MKDMBF$; MKI$; MKL$; MKS$; MKSMBF$

Funzione CVI

Descrizione

Trasforma una stringa di due byte (creata con MKI$) in un valore intero.

Sintassi

CVI(*stringa_due_byte*)

Esempio

Vedi CVD

Funzioni correlate: CVI; CVDMBF; CVL; CVS; CVSMBF; MKD$; MKDMBF$; MKI$; MKL$; MKS$; MKSMBF$

Funzione CVL

Descrizione

Trasforma una stringa di 4 byte in un valore intero di tipo LONG.

Sintassi

CVL(*stringa_quattro_byte*)

Esempio

Vedi CVD

Funzioni correlate = CVD; CVDMBF; CVI; CVS; CVSMBF;MKD$; MKDMBF$; MKI$; MKL$; MKS$; MKSMBF$

Funzione CVS

Descrizione

Trasforma una stringa di 4 byte (creata con MKS$) in un valore a precisione singola.

Sintassi

CVS(*stringa_quattro_byte*)

Esempio

Vedi CVD

Funzioni correlate: CVD; CVDMBF; CVI; CVL; CVSMBF; MKD$; MKDMBF$; MKI$; MKL$; MKS$; MKSMBF$

Funzione CVSMBF

Descrizione

Trasforma una stringa a 4 byte contenente un valore a precisione singola (creati da MKSMBF$) da un formato binario Microsoft a un formato IEEE.

Sintassi

CVSMBF(*stringa_quattro_byte*)

Note

- CVDMBF, CVSMBF, MKDMBF$ e MKSMBF$ sono utili per mantenere i file di dati creati con le versioni precedenti di Basic.

Esempio

Vedi CVD

Funzioni correlate: CVD, CVDMBF; CVI; CVL; CVS;MKD$; MKDMBF$; MKI$; MKL$; MKS$; MKSMBF$

Istruzione DATA

Descrizione

Registra costanti numeriche e stringhe che verranno lette dall'istruzione READ successiva.

Sintassi

DATA *costante* [,*costante*]...

Note

- *costante* è una costante di stringa o numerica.
- Le istruzioni READ accedono alle istruzioni DATA nell'ordine in cui le istruzioni DATA appaiono nel programma. L'istruzione RESTORE permette al programma di rileggere l'istruzione DATA se necessario.
- Il tipo di variabile nell'istruzione READ deve corrispondere al tipo di costante nell'istruzione DATA.

- Le istruzioni DATA possono essere utilizzate solo a livello di modulo, mai durante la procedura.

Esempio

```
DATA 1, 2.2345, "TEST", 98765
READ a%, b#, c$, d$
PRINT a%, b#, c$, d$
```

Se si lancia questo programma, viene visualizzato quanto segue:

```
1   2.2345   TEST   98765
```

Funzioni correlate: READ; STORE

Funzione DATE$

Descrizione

Riporta una stringa di 10 caratteri contenente la data di sistema corrente nel formato *mm-gg-aaaa*.

Sintassi

DATE$

Esempio

```
PRINT DATE$
```

Se si lancia questo programma il 13 gennaio 1991, viene visualizzato quanto segue:

```
01-13-1991
```

Funzioni correlate: DATE$; TIME$

Istruzione DATE$

Descrizione

Imposta la data di sistema.

Sintassi

DATE$ = *stringa*

Note

- *stringa* corrisponde a un'espressione contenente la data desiderata nel formato "*mm-gg-aaaa*" dove *mm* corrisponde al mese (da 1 a 12), *gg* al giorno (da 1 a 31) e *aaaa* corrisponde all'anno (dal 1980 al 2099).
- Se si specificano solo le ultime due cifre dell'anno, DATE$ suppone che le prime due siano 19.
- DATE$ consente di utilizzare sia i trattini (-) che le barre (/) per separare i campi delle date.

Esempio

```
DATE$ = "25-12-89"
```

Funzioni correlate: DATE$; TIME$

Istruzione DECLARE

Descrizione

Dichiara una procedura e indica a QBasic di eseguire un controllo della digitazione per ciascun parametro.

Sintassi

DECLARE {**FUNCTION** | **SUB**} *nome_procedura* [([*elenco_argomenti*])]

Note

- DECLARE indica a QBasic di controllare che tutti i tipi di parametri passati a una procedura corrispondano a quelli necessari per quella procedura stessa.
- DECLARE è necessario solo quando non si utilizza la parola chiave CALL.
- La parola chiave FUNCTION indica che la procedura è una funzione, nello stesso modo, SUB indica un sottoprogramma.
- *nome_procedura* corrisponde al nome della funzione o del sottoprogramma.
- *elenco_argomenti* corrisponde a una lista di parametri facoltativa, in cui i parametri sono separati tra loro da virgole.

Per il controllo del compilatore si specifichi l'argomento come segue:
nome_variabile [**AS** *tipo*]

- Tipi accettabili sono: INTEGER, LONG, SINGLE, DOUBLE, STRING, ANY o un tipo definito dall'utente. ANY consente di utilizzare qualsiasi tipo per il parametro.

Esempio

Vedi CALL

Istruzioni correlate: CALL; FUNCTION; SUB

Istruzioni DEF FN

Descrizione

Definisce una funzione.

Sintassi

DEF FN*nome* [(*elenco_argomenti*)] = *espressione*
oppure
DEF FN*nome* [(*elenco_argomenti*)]
...
FN*nome* = *espressione*
...
END DEF

Note

- I nomi delle funzioni devono iniziare con FN e possono contenere fino a 40 caratteri. Il nome della funzione indica il tipo di valore che la funzione riporta:

Nome funzione	**Riporto**
FNday$	stringa
FNcount%	intero
FNaverage#	valore a precisione singola

- Perché una funzione riporti un valore, questa deve assegnare il risultato al nome della funzione.

- *elenco_argomenti* corrisponde a un elenco di parametri relativi alla funzione tra loro separati da virgole come segue:
nomi_argomenti [**AS** *tipo*]

- Non è possibile utilizzare una funzione prima che il programma l'abbia definita e neppure utilizzare più volte la funzione DEF FN.

Esempio

```
DEF FNnum (a AS INTEGER, b AS INTEGER) = a + b
DEF FNmax (a AS INTEGER, b AS INTEGER, c AS INTEGER)
    IF (a > b) THEN
max = a
ELSE
max = b
END IF
IF (max > c) THEN
FNmax = max
ELSE
FNmax = c
END IF
END DEF

PRINT FNsum(3, 5)
PRINT FNmax(1, 2, 3)
```

Se si lancia questo programma, verrà visualizzato quanto segue:

```
8
3
```

Istruzioni correlate: EXIT; FUNCTION

Istruzione DEF SEG

Descrizione

Imposta l'indirizzo del segmento corrente per funzioni PEEK successive e per le istruzioni BLOAD, CALL ABSOLUTE e POKE.

Sintassi

DEF SEG [= *indirizzo*]

Note

- *indirizzo* corrisponde a un'espressione intera che può variare da 0 a 65.535. Se si omette *indirizzo*, il QBasic utilizza il segmento dati Basic.

Esempio

Vedi istruzione CALL ABSOLUTE

Funzioni correlate: PEEK

Istruzioni correlate: BLOAD; BSAVE; CALL ABSOLUTE; POKE

Istruzione DEFDBL

Descrizione

Definisce il tipo di dati di default a doppia precisione per variabili i cui nomi inizino con una lettera nell'intervallo specificato.

Sintassi

DEFDBL *lettera*[-*ultima_lettera*][,*lettera*[-*ultima_lettera*]]

Note

- *lettera* e *ultima_lettera* corrispondono a un intervallo di lettere associate a un tipo. Il QBasic non distingue tra variabili in formato maiuscolo o minuscolo.
- Se il nome di una variabile ha il suffisso *%, &, !, #* o *$*, i tipi di dati associati al suffisso hanno la precedenza sull'istruzione del tipo di default.

Esempio

```
DEFDBL A-J
```

Istruzioni correlate: DEFINT; DEFLNG; DEFSNG; DEFSTR

Istruzione DEFINT

Descrizione

Definisce il tipo di dati di default INTEGER per variabili i cui nomi inizino con una lettera nell'intervallo specificato.

Sintassi

DEFINT *lettera*[-*ultima_lettera*][, *lettera*[-*ultima-lettera*]]...

Note

Vedi DEFDBL

Esempio

```
DEFINT X-Z
```

Istruzioni correlate: DEFDBL; DEFLNG; DEFSNG; DEFSTR

Istruzione DEFLNG

Descrizione

Definisce il tipo dei dati di default LONG per variabili il cui nome inizi con una lettera nell'intervallo specificato.

Sintassi

DEFLNG *lettera*[*-ultima_lettera*][,*lettera*[*-ultima_lettera*]]...

Note

Vedi DEFDBL

Esempio

```
DEFLNG L-N
```

Istruzioni correlate: DEFDBL; DEFINT; DEFSNG; DEFSTR.

Istruzione DEFSNG

Descrizione

Definisce il tipo di dati di default a precisione singola per variabili il cui nome inizi con una lettera all'interno dell'intervallo specificato.

Sintassi

DEFSNG *lettera*[*-ultima_lettera*][, *lettera*[*-ultima_lettera*]]...

Note

Vedi DEFDBL

Esempio

```
DEFSNG T-W
```

Istruzioni correlate: DEFDBL; DEFINT; DEFLNG; DEFSTR

Istruzione DEFSTR

Descrizione

Definisce il tipo di dati di default come stringa per variabili il cui nome inizi con una lettera nell'intervallo specificato.

Sintassi

DEFSTR *lettera*[-*ultima_lettera*][, *lettera*[-*ultima_lettera*]]...

Note

Vedi DEFDBL

Esempio

```
DEFSTR S
'sdate viene portato per default al formato stringa
sdata = DATE$
PRINT sdate
```

Istruzioni correlate: DEFDBL; DEFINT; DEFLNG; DEFSNG

Istruzione DIM

Descrizione

Dichiara una variabile di array e ne memorizza la posizione.

Sintassi

DIM[**SHARED**] *nome_variabile* [(*indice_inf*)][**AS** *tipo*][,*nome_variabile* [(*indice_inf*)]][**AS** *tipo*]]...

Note

- La parola chiave SHARED consente ai sottoprogrammi e alle funzioni di dividere la stessa variabile senza passarla come un parametro.
- *nome_variabile* corrisponde al nome dell'array.
- *indice_inf* corrisponde all'indice inferiore delle dimensioni dell'array. Se in un'istruzione DIM precedente non fosse presente un array, sarà possibile assegnare un valore a un elemento il cui valore sia contenuto nell'intervallo da 0 a 10. Il limite superiore e inferiore può essere modificato come mostrato:

```
DIM a(0 TO 8)    'a(0) to a(8)
DIM b(1 TO 10)   'b(1) to b(10)
```

- Se si specifica solo un indice inferiore, il QBasic suppone che sia il limite superiore e utilizza 0 come limite inferiore a meno che non venga inclusa un'istruzione OPTION BASE.
- Per gli array multidimensionali, sarà sufficiente separare gli indici inferiori della dimensione di ciascun array con delle virgole:

```
DIM box(3, 3)
DIM bigbox(1 TO 10, 1 TO 10)
```

- La dimensione massima di un array è 60.
- Tipi accettabili di array comprendono INTEGER, LONG, SINGLE, DOUBLE, STRING e i tipi definiti dall'utente.

Esempio

```
DIM a(25 TO 100) AS INTEGER
DIM b(1 TO 10, 1 TO 5) AS DOUBLE

TYPE Scheda
   giorno AS STRING * 10
   ore AS INTEGER
END TYPE
DIM lavorativi AS Scheda
```

Funzioni correlate: LBOUND; UBOUND

Istruzioni correlate: ERASE; OPTIONBASE; REDIM

Istruzione DO UNTIL

Descrizione

Ripete una serie di istruzioni fino a quando non si verifica una condizione.

Sintassi

DO UNTIL *espressione_booleana*
istruzioni
LOOP
oppure
DO
istruzioni
LOOP UNTIL *espressione_booleana*

Note

- *espressione_booleana* è un'espressione che verifica se una condizione viene soddisfatta o meno, come per esempio I>100.
- La prima forma dell'istruzione controlla prima l'espressione booleana. Se l'espressione è verificata, il QBasic salta le istruzioni all'interno del loop e prosegue l'esecuzione con la prima istruzione dopo il loop stesso. Se l'espressione è falsa, il QBasic esegue l'istruzione all'interno del loop fino a quando la condizione non viene verificata.
- La seconda forma dell'istruzione esegue prima l'istruzione all'interno del loop e quindi verifica l'espressione booleana. Se l'espressione è falsa, il QBasic ripete l'istruzione del loop. Se l'espressione è vera, il QBasic prosegue l'esecuzione partendo dalla prima istruzione subito dopo il loop.

Esempio

```
i = 0
DO
   PRINT i
   i = i + 1
LOOP UNTIL i = 100
```

Istruzioni correlate: DO WHILE; EXIT; FOR; WHILE/WEND

Istruzione DO WHILE

Descrizione

Ripete una serie di istruzioni mentre una condizione è vera.

Sintassi

DO WHILE *espressione_booleana*
  istruzioni
LOOP
oppure
DO
  istruzioni
LOOP WHILE *espressione_booleana*

Note

- *espressione_booleana* è un'espressione che verifica se una condizione è vera o falsa.
- La prima forma dell'istruzione verifica l'espressione booleana. Se questa è vera, il QBasic esegue le istruzioni all'interno del loop fino a quando l'espressione non diventa falsa. Una volta che l'espressione è diventata falsa, il programma prosegue l'esecuzione partendo dala prima istruzione dopo il loop.
- La seconda forma esegue prima l'istruzione all'interno del loop e quindi verifica l'espressione booleana. Se l'espressione è vera, il QBasic ripete le istruzioni; altrimenti l'esecuzione continua partendo dalla prima istruzione dopo il loop.

Esempio

```
i = 0
DO WHILE i < 100
    PRINT i
i = i + 1
LOOP
```

Istruzioni correlate: DO UNTIL; EXIT; FOR; WHILE/WEND

Istruzione DRAW

Descrizione

Disegna un oggetto specificato in una stringa.

Sintassi

DRAW *stringa*

Note

- DRAW utilizza una stringa contenente comandi grafici per disegnare oggetti. La stringa può contenere comandi per il movimento del cursore, per il colore e per il ridimensionamento del disegno.
- I comandi per il movimento del cursore sono i seguenti:

Comando	Descrizione
U [*n*]	In alto di *n* spazi.
D [*n*]	In basso di *n* spazi.
L [*n*]	A sinistra di *n* spazi.
R [*n*]	A destra di *n* spazi.
E [*n*]	In alto e a destra di *n* spazi.
F [*n*]	In basso e a destra di *n* spazi.
G [*n*]	In basso e a sinistra di *n* spazi.
H [*n*]	In alto e a sinistra di *n* spazi.
M [{–|+}]*x,y*	Sposta verso *x,y* (se *x,y* vengono preceduti da un segno + o – il movimento è relativo alla posizione corrente del cursore).

Se si omette *n*, il cursore si sposta di uno spazio.

- DRAW consente di far precedere le funzioni del movimento del cursore con B e N:

Comando	Descrizione
B	Sposta il cursore ma non disegna punti.
N	Sposta il cursore ma lo restituisce alla posizione originale una volta terminato il disegno.

- I comandi per il colore, l'angolo e il ridimensionamento sono i seguenti:

Comando	Descrizione
A *rotazione*	Imposta la rotazione dell'angolo in gradi: 0 = 0, 1 = 90, 2 = 180, 3 = 270
TA *gradi*	Gira un angolo (da –360 a 360 gradi)
C *colore*	Imposta il colore
S *n*	Imposta il fattore di scala per spazi (il valore di default è 4, cioè 1 pixel).
P *colore, margine*	Colora l'interno dell'oggetto con *colore*; il colore desiderato per il margine sarà *margine*.

Esempio

```
'Disegna un quadrato e lo riempie:
SCREEN 1
DRAW "C3"
DRAW "L20U20R20D20"     'disegna il quadrato
DRAW "BH10"            'sposta il cursore nel quadrato
DRAW "P2,3"            'colora il quadrato
```

Funzioni correlate: POINT

Istruzioni correlate: CIRCLE; COLOR; LINE; PAINT; PRESET; PSET; SCREEN

Istruzione END

Descrizione

Termina il programma QBasic.

Sintassi

END

Note

- END termina il programma e chiude tutti i file.
- L'istruzione END viene inoltre utilizzata con DEF, FUNCTION, IF, SELECT, SUB e TYPE. Queste forme verranno discusse nei paragrafi sulle relative istruzioni.

Esempio

```
FOR i = 1 TO 10
    PRINT i
NEXT i
END
```

Istruzioni correlate: DEF FN; FUNCTION; IF; SELECT CASE, SUB; TYPE.

Istruzione ENVIRON

Descrizione

Modifica una voce esistente o inserisce una nuova voce nell'ambiente MS-DOS.

Sintassi

ENVIRON *stringa*

Note

- L'istruzione ENVIRON aspetta un'espressione della stessa forma del comando MS-DOS SET *voce=valore*
- La modifica della tabella di ambiente è valida solo per la durata del programma.
- Non è possibile aumentare le dimensioni dell'ambiente MS-DOS in QBasic. Per creare uno spazio ambiente MS-DOS e utilizzare i programmi QBasic, si crei una voce vuota con il comando DOS Set. Quindi si cancelli il contenuto della voce in un programma QBasic per creare spazio per variabili nuove o modificate.

Esempio

```
ENVIRON "PROGRAM=TEST"
PRINT ENVIRON$("PROGRAM")
```

Se si lancia questo programma, viene visualizzato quanto segue:

```
TEST
```

Funzioni correlate: ENVIRON$

Funzione ENVIRON$

Descrizione

Restituisce una voce dall'ambiente MS-DOS

Sintassi

ENVIRON$(*stringa_voce*)
oppure
ENVIRON$(*n*)

Note

- Il comando MS-DOS Set permette di impostare e visualizzare stringhe dal prompt dell'MS-DOS.
- La prima forma di ENVIRON$ permette al programma di accedere al valore di una variabile d'ambiente. Il nome della variabile desiderata è *stringa_voce*.
- La seconda forma di ENVIRON$ permette al programma di accedere alla ennesima stringa d'ambiente.
- Se la voce specificata non esiste, ENVIRON$ restituisce una stringa vuota.

Esempio

```
PRINT ENVIRON$("PATH")

i = 1
DO WHILE ENVIRON$(I) <> ""
PRINT ENVIRON$(I)
i = i + 1
LOOP
```

Istruzioni correlate: ENVIRON

Funzione EOF

Descrizione

Verifica la condizione di fine file.

Sintassi

EOF(*numero_file*)

- EOF restituisce un valore affermativo se è stata raggiunta la fine del file associato al numero di file specificato; altrimenti, EOF restituisce un valore falso.
- *numero_file* corrisponde al numero assegnato al file nell'istruzione OPEN

Esempio

```
OPEN "\CONFIG,SYS" FOR INPUT AS #1
DO UNTIL EOF(1)
   LINE INPUT #1, FDATA$
   PRINT FDATA$
LOOP
CLOSE #1
```

Funzioni correlate: LOC; LOF

Istruzioni correlate: CLOSE; OPEN

Istruzione ERASE

Descrizione

Reinizializza gli elementi di un array statico o libera array dinamici.

Sintassi

ERASE *array*[,*array*]...

Note

- *array* corrisponde al nome dell'array da reinizializzare o liberare.
- Per gli array numerici statici, ERASE imposta ciascun elemento a zero. Per gli array stringa statici, ERASE imposta ciascun elemento a null.
- Per gli array dinamici, ERASE libera la memoria utilizzata dagli array specificati.

Esempio

```
DIM a(100)
FOR i = 1 TO 100
        a(i) =i
```

```
NEXT i
ERASE a        'Reinizializza A
FOR i = 1 TO 100
     PRINT a(i)
NEXT i
```

Funzioni correlate: FRE

Istruzioni correlate: CLEAR; DIM; REDIM

Funzione ERDEV

Descrizione

Restituisce un codice di errore di tipo INTEGER dall'ultima periferica che ha dichiarato un errore.

Sintassi

ERDEV

Note

- Il gestore di errori critici dell'MS-DOS imposta il valore per ERDEV. Il byte più basso contiene il codice di errore di MS-DOS (da 0 a 12). Il byte più alto contiene informazioni riguardo all'attributo di periferica.

Esempio

```
ON ERROR GOTO handler
...
handler:
    PRINT "Errore nell'accesso alla periferica "; ERDEV$
    PRINT "Errore nel codice di stato;" ERDEV
...
```

Funzioni correlate: ERDEV$; ERL; ERR

Istruzioni correlate: ERROR; ON ERROR GOTO; RESUME

Funzione ERDEV$

Descrizione

Restituisce una stringa di caratteri contenente il nome della periferica che ha generato l'errore critico.

Sintassi

ERDEV$

Note

- Il gestore di errori critici (handler) dell'MS-DOS imposta il valore per ERDEV$.

Esempio

Vedi ERDEV

Funzioni correlate: ERDEV; ERL; ERR

Istruzioni correlate: ERROR; ON ERROR GOTO; RESUME

Funzione ERL

Descrizione

Restituisce il numero di riga dell'istruzione che ha causato l'errore o il numero della riga più vicina prima dell'istruzione che ha causato l'errore.

Sintassi

ERL

Note

- ERL restituisce solo i numeri di riga. Non restituisce le righe di etichetta. Se non si stessero utilizzando i numeri di riga, ERL restituirebbe 0.

Esempio

```
100 PRINT 1/0
1000 Handler:
1010        PRINT "Errore di procedura alla riga"; ERL
```

```
1020        PRINT "Numero errore"; ERR
1030        RESUME
```

Se si lancia questo programma, viene visualizzato quanto segue:

```
Errore di procedura alla riga 100
Numero errore 11
```

Funzioni correlate: ERDEV; ERDEV$; ERR

Istruzioni correlate: ERROR; ON ERROR GOTO; RESUME

Funzione ERR

Descrizione

Restituisce il codice di errore per l'ultimo errore verificatosi.

Sintassi

ERR

Note

Codice errore	Descrizione
1.	NEXT senza FOR.
2.	Errore di sintassi.
3.	RETURN senza GOSUB.
4.	Valori istruzioni DATA esauriti.
5.	Chiamata di funzione non valida.
6.	Overflow.
7.	Memoria esaurita.
8.	Etichetta non definita.
9.	Indice inferiore fuori limite.
10.	Definizione doppia.
11.	Divisione per zero
12.	Non ammesso in modalità diretta.
13.	Errore di digitazione.

Codice errore	Descrizione
14.	Spazio stringa esaurito.
16.	Formula a stringa troppo complessa.
17.	Impossibile proseguire.
18.	Funzione non definita.
19.	Manca RESUME.
20.	RESUME senza errore.
24.	Timeout di periferica.
25.	Errore di periferica.
26.	FOR senza NEXT.
27.	Carta esaurita.
29.	WHILE senza WEND.
30.	WEND senza WHILE.
33.	Etichetta doppia.
35.	Sottoprogramma non definito.
37.	Numero argomenti non corrispondente.
38.	Array non definito.
39.	Atteso CASE ELSE.
40.	Richiesta variabile.
50.	Overflow nell'istruzione FIELD.
51.	Errore interno.
52.	Nome o numero file errato.
53.	File non trovato.
54.	Modalità di accesso al file errata.
55.	File già aperto.
56.	Istruzione FIELD attivata.
57.	Errore I/O su periferica.
58.	File già esistente.
59.	Lunghezza record errata.

Codice errore	Descrizione
61.	Disco pieno.
62.	Input oltre la fine del file.
63.	Numero di record errato.
64.	Nome file errato.
67.	Troppi file.
68.	Periferica non disponibile.
69.	Overflow del buffer comunicazioni.
70.	Permesso negato.
71.	Disco non pronto.
72.	Errore di supporto del disco.
73.	Funzione non disponibile.
74.	Tentativo di rinominare il disco.
75.	Errore di accesso al file/percorso.
76.	Percorso non trovato.

Esempio

Vedi ERL

Funzioni correlate: ERDEV; ERDEV$; ERL

Istruzioni correlate: ERROR; ON ERROR GOTO; RESUME

Istruzione ERROR

Descrizione

Simula il verificarsi del numero di errore specificato. Consente a un programma di definire i propri codici di errore.

Sintassi

ERROR *espressione_numerica*

Note

- *espressione_numerica* corrisponde a un valore intero nell'intervallo tra 1 e 255.
- Si veda la funzione ERR per un elenco dei codici di stato di valore predefiniti.
- L'istruzione ERROR assegna il valore dell'errore specificato a ERR e passa il controllo al gestore di errore.

Esempio

```
ON ERROR GOTO HANDLER
'Controlla il gestore di errore tramite errore 222
ERROR 222
EndTest:
END

Handler:
   PRINT "Controlla il gestore dell'errore"
PRINT " tramite errore 222"; ERR
   RESUME EndTest
```

Funzioni correlate: ERDEV; ERDEV$; ERL; ERR

Istruzioni correlate: ON ERROR GOTO; RESUME

Istruzione EXIT

Descrizione

Esce da un loop di DO o FOR, una funzione o un sottoprogramma.

Sintassi

EXIT DEF Esce da una funzione DEF FN
oppure
EXIT DO Esce da un loop DO
oppure
EXIT FOR Esce da un loop FOR
oppure
EXIT FUNCTION Esce da una procedura FUNCTION
oppure
EXIT SUB Esce da un sottoprogramma

Note

- Per i loop DO e FOR, l'esecuzione continua a parire dalla prima istruzione dopo il loop.
- Per le funzioni e i sottoprogrammi l'esecuzione prosegue dall'istruzione successiva a quella che ha chiamato la funzione o la subroutine.

Esempio

```
j = 30
FOR i = 1 TO 50
        IF i = j THEN
           EXIT FOR
        END IF
NEXT i
PRINT "Il valore finale è"; i
```

Se si lancia questo programma, viene visualizzato quanto segue:

```
Il valore finale è 30.
```

Istruzioni correlate: DEF FN; DO UNTIL; DO WHILE; FOR; FUNCTION; SUB

Funzione EXP

Descrizione

Restituisce *e* elevato a una potenza specificata, dove *e* corrisponde alla base di un logaritmo naturale.

Sintassi

EXP(*espressione_numerica*)

Note

- *espressione_numerica* specifica la potenza a cui *e* deve essere elevato. Deve essere minore o uguale a 88,02969; altrimenti EXP restituisce un errore di overflow.

Esempio

```
PRINT EXP(0), EXP(1)
```

Se si lancia questo programma, viene visualizzato quanto segue:

```
1       2.71828
```

Funzioni correlate: LOG

Istruzione FIELD

Descrizione

Alloca spazio per variabili in un buffer ad accesso casuale.

Sintassi

FIELD [#]*numero_file*, *larghezza* **AS** *variabile*

Note

- *numero_file* corrisponde al numero assegnato al file nella relativa istruzione OPEN.
- *larghezza* corrisponde al numero di caratteri nel campo.
- *variabile* corrisponde al nome della variabile che deve essere utilizzata durante la lettura o la scrittura da o su file.
- Il nome di una variabile che appare nell'istruzione FIELD non dovrebbe apparire in un'istruzione INPUT o sul lato sinistro di un operatore di assegnazione. Se apparisse, la variabile non si riferirebbe più al buffer di file ad accesso casuale.
- Utilizzare le variabili record è più pratico rispetto all'utilizzo dell'istruzione FIELD.

Esempio

```
OPEN "RANDOM.DAT" FOR RANDOM AS #1 LEN=80
FIELD #1, 76 AS nome$, 4 AS paghe$
```

Istruzioni correlate: GET; LSET; OPEN; PUT; RSET

Funzione FILEATTR

Descrizione

Restituisce il valore di modalità del file o il gestore MS-DOS per un file aperto.

Sintassi

FILEATTR(*numero_file*, *info_file*)

Note

- *numero_file* corrisponde al numero di file assegnato al file nella sua istruzione OPEN.
- *info_file* Specifica il tipo di informazioni che si desidera vengano restituite. Se il valore di *info_file* corrisponde a 1, FILEATTR restituisce un valore che indica la modalità di accesso del file.

Valore	Modo
1	Input
2	Output
3	Random
8	Append
32	Binario

- Se il valore di *info_file* corrisponde a 2, FILEATTR restituisce il gestore di file del DOS.

Esempio

```
OPEN "APPEND.DAT" FOR APPEND AS #1
OPEN "OUTPUT.DAT" FOR OUTPUT AS #2

PRINT "File 1 Mode": FILEATTR(1, 1)
PRINT "File 1 Handle"; FILEATTR(1, 2)
PRINT "File 2 Mode"; FILEATTR(2, 1)
PRINT "File 2 Handle"; FILEATTR(2, 2)
CLOSE #1: CLOSE #2
```

Se si lancia questo programma, viene visualizzato quanto segue

```
File 1 Mode 8
File 1 Handle 5
File 2 Mode 2
File 2 Handle 6
```

Istruzione FILES

Descrizione

Visualizza i nomi dei file nella directory specificata o corrente.

Sintassi

FILES [*stringa*]

Note

- *stringa* corrisponde a un'espressione che contiene una specificazione file MS-DOS dei file da visualizzare. È consentito l'utilizzo di caratteri jolly.
- Se si omette la specificazione di un file, FILES visualizza i file nella directory corrente.

Esempio

```
FILES                   'Elenca tutti i file
FILES "*.BAS"            'Elenca tutti i file .BAS
FILES "A:"              'Elenca tutti i file nell'unità A:
```

Istruzioni correlate: CHDIR; KILL; NAME

Funzione FIX

Descrizione

Restituisce la porzione intera di un'espressione con virgola mobile.

Sintassi

FIX(*espressione_numerica*)

Note

- *espressione_numerica* corrisponde a qualsiasi espressione numerica.

Esempio

```
PRINT FIX(-10.99)
PRINT FIX(-10.1)
```

Questo programma produce la seguente visualizzazione:

```
-10
-10
```

Istruzioni correlate: INT

Istruzione FOR

Descrizione

Ripete una serie di istruzioni date per un determinato numero di volte.

Sintassi

FOR *variabile_controllo* = *valore_inizio* **TO** *valore_finale* [**STEP** *increment*]

...

NEXT [*variabile_controllo*[,*variabile_controllo*]...]

Note

- *variabile_controllo* corrisponde alla variabile che FOR aumenta con ciascuna iterazione del loop. Controlla se il QBasic ripete il loop o meno.
- *valore_inizio* corrisponde al valore iniziale che QBasic assegna alla variabile di controllo.
- *valore_finale* corrisponde al valore che la variabile di controllo deve assumere prima che il loop termini.
- *increment* corrisponde alla cifra che QBasic deve aggiungere alla variabile di controllo con ciascuna iterazione del loop. L'aumento può essere una valore positivo o negativo. Se si omette *increment* l'aumento di default è di 1.
- L'istruzione NEXT indica a QBasic di aumentare il controllo di una variabile e di controllare se è maggiore del valore finale. Se non lo fosse, l'esecuzione proseguirebbe dalla prima istruzione all'interno del loop; altrimenti l'esecuzione continuerebbe dalla prima istruzione successiva a NEXT.

Esempio

```
FOR i = 1 TO 10
        PRINT "i ="; i
NEXTi
FOR i = 1 TO 10
     FOR j = 1 TO 10
          PRINT i; "*"; j; "="; i*j
```

```
    NEXT j
NEXT i
```

Istruzioni correlate: DO UNTIL; DO WHILE; EXIT; WHILE/WEND

Istruzione FRE

Descrizione

Restituisce la quantità di spazio disponibile, lo spazio di stringa e di memoria.

Sintassi

FRE(*espressione_numerica*)

oppure

FRE(*stringa*)

Note

- Se l'argomento di FRE è 1, la funzione restituisce le dimensioni in byte del più largo array che è possibile creare. Se l'argomento è 2, FRE restituisce lo spazio di stack disponibile. Per qualsiasi altro argomento numerico, FRE restituisce la quantità di spazio stringa disponibile.
- Se l'argomento di FRE è una stringa, la funzione compatta lo spazio libero di stringa in un blocco singolo e restituisce lo spazio di stringa disponibile.

Esempio

```
PRINT "Spazio stringa"; FRE("")
PRINT "Spazio di stack"; FRE(-2)
PRINT "Spazio di array"; FRE(-1)
```

Se si lancia questo programma, viene visualizzato, per esempio, quanto segue:

```
Spazio stringa 48460
Spazio stack 784
Spazio array 184092
```

Istruzione correlata: CLEAR; ERASE

Funzione FREEFILE

Descrizione

Restituisce il numero di file Basic successivo disponibile.

Sintassi

FREEFILE

Note

- FREEFILE elimina la necessità di codificare pesantemente i numeri di file e quindi il rischio di utilizzare un numero di file già in uso.

Esempio

```
numero_file = FREEFILE
OPEN "TEST.DAT" FOR OUTPUT AS numero_file
CLOSE numero_file
```

Istruzione correlata: OPEN

Istruzione FUNCTION

Descrizione

Dichiara una funzione definita dall'utente.

Sintassi

FUNCTION *nome_funzione*[(*argomenti*)][**STATIC**]
...
nome_funzione = *espressione*
...
END FUNCTION

Note

- *nome_funzione* corrisponde al nome della funzione definita dall'utente. Il nome può terminare con un carattere di dichiarazione di tipo (*%*, *&*, *!*, o *$*) per indicare il tipo di valore che restituisce.

- *argomento* corrisponde a una lista di parametri opzionali, separati tra loro da virgole per essere passati alla funzione.

- Per specificare il tipo di ciascuna variabile, si utilizzi il seguente formato: *variabile*[()] **AS** *tipo*
- La parola chiave STATIC indica a QBasic di salvare i valori delle variabili locali di funzione tra le chiamate di funzione.
- Perché una funzione restituisca un valore, questa deve assegnare un'espressione al nome della funzione in qualche punto.

Esempio

```
FUNCTION Min% (a AS INTEGER, b AS INTEGER)
    IF (a < b) then
           Min% = a
    ELSE
           Min% = b
    END IF
END FUNCTION

PRINT "Min di 5 e 3 è"; Min%(5, 3)
```

Istruzione correlata: DECLARE; DEF FN; EXIT; STATIC; SUB

Istruzione GET (File I/O)

Descrizione

Legge un record dal file su disco binario o ad accesso casuale.

Sintassi

GET [#]*numero_file*[,[*numero record*][, *variabile*]]

Note

- *numero_file* corrisponde al numero assegnato al file nella relativa istruzione OPEN.
- *numero_record* corrisponde al numero di record desiderato da 1 a 2.147.483.647. Se si omette il numero di record, GET legge il record successivo.
- *variabile* corrisponde al nome della variabile in cui GET immette i dati. Di solito, viene utilizzata una variabile record definita dall'utente.

Esempio

```
TYPE RecordStipendi
      dnome AS STRING * 20
paga AS SINGLE
END TYPE
DIM dipendente AS RecordStipendi

OPEN "STIPENDI.DAT" FOR RANDOM AS #1 LEN = LEN(dipendente)
GET#1, 1, dipendente
PRINT dipendente.dnome, dipendente.paga
CLOSE #1
```

Funzione correlata: CVD; CVI; CVL; CVS; MKD$; MKI$; MKL$; MKS$

Istruzione correlata: FIELD; INPUT; LINE INPUT; LSET; PUT; RSET

Istruzione GET (Graphics)

Descrizione

Registra un'immagine grafica in un array.

Sintassi

GET [**STEP**](*xsinistra*, *yalto*)-[**STEP**](*xdestra*, *ybasso*), *array*[(*indice*)]

Note

- GET registra l'immagine contenuta nel rettangolo specificato.
- La parola chiave STEP indica che le coordinate sono offset relativi all'ultimo punto tracciato.
- *array* corrisponde al nome dell'array in cui GET deve registrare l'immagine.
- *indice* corrisponde all'indice di array da cui inizia la memorizzazione dell'immagine grafica.
- Per determinare il numero di byte richiesti, si utilizzi la seguente formula: 4 + INT(((*xdestra* – *xsinistra* + 1) * (*bit_per_pixel*/*piani*) + 7)/8) * *piani* * (*ybasso* – *yalto* + 1)
- I numeri di bit per pixel e il numero di *piani* dipendono dal video attivo:

Modalità video	Bit per pixel	Piani
1	2	1
2	1	1
3	1	1
4	1	1
7	1	4
8	1	4
9	1	*
10	1	2
11	1	1
12	1	4
13	8	1

** 2 se ci sono 64 KB di memoria EGA; altrimenti 4.*

Esempio

```
GET (10, 20) - (20, 50), animale
```

Istruzioni correlate: PUT (Graphics); SREEN

Istruzione GOSUB

Descrizione

Indica di proseguire l'esecuzione da una subroutine Basic.

Sintassi

GOSUB *locazione*

Note

- *locazione* corrisponde sia a un numero di riga sia a un'etichetta da cui l'esecuzione può proseguire.
- Si utilizzi un'istruzione RETURN per concludere la subroutine.
- GOSUB è il metodo più vecchio per accedere alle subroutine. La maggior parte dei nuovi programmi utilizzano le istruzioni QBasic SUB e CALL.

Esempio

```
GOSUB Test
END
Test:
    PRINT "Test in subroutine"
RETURN
```

Istruzioni correlate: ON operazione GOSUB; ON *espressione*; RETURN; SUB

Istruzione GOTO

Descrizione

Salta al numero di riga o all'etichetta specificata.

Sintassi

GOTO *locazione*

Note

- *locazione* corrisponde al numero di riga o all'etichetta da cui prosegue l'esecuzione.
- Le versioni precedenti di Basic non presentavano i loop DO, i casi ELSE per l'istruzione IF o le istruzioni SELECT CASE e utilizzavano GOTO per completare queste costruzioni. Per migliorare la leggibilità del programma e semplificare il debugging, si limiti l'uso di GOTO.

Esempio

```
i = 0
Inizio:
    PRINT "i ="; i
INPUT "Di nuovo"; risposta$
IF risposta$="N" THEN
     END
ELSE
    i = i + 1
     END IF
GOTO Inizio
```

Istruzioni correlate: DO UNTIL; DO WHILE; IF; SELECT CASE

Funzione HEX$

Descrizione

Restituisce una stringa di caratteri contenente la rappresentazione esadecimale di un valore.

Sintassi

HEX$(*espressione_numerica*)

Note

- La notazione esadecimale è la base di un sistema di numerazione a 16 unità. Utilizza i numeri da 1 a 9 e le lettere da A a F. Per registrare la rappresentazione esadecimale di un numero, sarà necessario utilizzare una variabile stringa.

Esempio

```
'Visualizza i valori ottali, decimali
'ed esadecimali da 0 a 255
FOR i = 0 TO 255
   PRINT OCT$(i), i, HEX$(i)
NEXT i
```

Funzioni correlate: OCT$

Istruzione IF

Descrizione

Fornisce un'esecuzione condizionale basata sulla verifica o meno di un'espressione.

Sintassi

IF *espressione* **THEN** *istruzioni_vere*[**ELSE** *istruzioni_false*]
oppure
IF *espressione* **THEN**
[*istruzioni_vere*]
[**ELSEIF** *espressione* **THEN**]
[*istruzioni_vere*]

...
[**ELSE**
  [*istruzioni_false*]]
END IF

Note

- La prima forma dell'istruzione permette di eseguire un'istruzione singola se l'espressione è vera e un'altra se l'istruzione è falsa.
- La seconda forma dell'istruzione permette di eseguire una serie di istruzioni se l'espressione è vera e un'altra serie se l'espressione è falsa. Inoltre questa sintassi permette di controllare una serie di diverse condizioni, una dopo l'altra.

Esempio

```
IF (a > b) THEN max = a ELSE max = b

IF (a > max) THEN CALL ValoreAlto(a)

IF (GIORNO$ = "LUNEDI") THEN
    CALL Riunioni
    CALL Cene
ELSE
    CALL Ginnastica
END IF

IF (GIORNO$ = "LUNEDI") THEN
    CALL Riunioni
ELSEIF (GIORNO$ = "MERCOLEDI") THEN
    CALL Biglietti
ELSEIF (GIORNO$ = "VENERDI") THEN
    CALL Cinema
END IF
```

Istruzioni correlate: ON *espressione*; SELECT CASE

Funzione INKEY$

Descrizione

Legge un carattere dalla tastiera.

Sintassi

INKEY$

Note

- INKEY$ restituisce una stringa nulla se nessun carattere è presente, una stringa da 1 byte per tasti standard e una stringa da 2 byte per i tasti estesi.
- Per i tasti estesi, il primo è un carattere 0 (ASCII 0) e il secondo corrisponde al codice scan della tastiera.
- INKEY$ non restituisce il carattere a video.

Esempio

```
PRINT "premere una serie di tasti - F10 per interrompere"
DO
      DO
            k$ = INKEY$
      LOOP WHILE k$ = ""
      IF LEN (k$) = 1 THEN
         PRINT "lettera", k$
      ELSE
         PRINT "Codice scan", ASC(MID$(k$, 2, 1))
      END IF
LOOP UNTIL MID$(k$, 2, 1) = CHR$(68)     'F10 codice scan
```

Funzione INP

Descrizione

Restituisce un byte letto da una porta I/O.

Sintassi

INP(*numero_porta*)

Note

- *numero_porta* corrisponde al numero associato alla porta desiderata. Deve essere compreso tra 0 e 65.535.

Esempio

```
'Attiva lo speaker attraverso la porta 97
salvaval = INP(97)
OUT 97, salvaval + 3
DO
LOOP WHILE INKEY$ = ""
OUT 97, salvaval
```

Istruzioni correlate: OUT

Istruzione INPUT

Descrizione

Accetta l'input da tastiera.

Sintassi

INPUT [;]["*prompt*"{;|,}]*variabili*

Note

- Un punto e virgola subito dopo INPUT indica al QBasic di lasciare il cursore sulla stessa riga dopo che l'utente avrà premuto Invio.
- *prompt* corrisponde al prompt facoltativo che INPUT visualizza.
- Un punto e virgola dopo il prompt indica a INPUT di visualizzare un punto di domanda dopo il prompt.

- Una virgola dopo il prompt indica a INPUT di eliminare il punto di domanda.
- *variabile* corrisponde all'elenco delle variabili da immettere. Le variabili dovranno essere separate con virgole.
- Se si immette un tipo di variabile diverso da quello che si dovrebbe immettere, o si immettono troppi o troppo pochi valori, INPUT visualizza il messaggio *Ripetere dall'inizio* e sarà necessario inserire nuovamente i dati.
- Più voci dovranno essere separate con virgole.

Esempio

```
'Punto di domanda
INPUT "Immettere il nome e l'età"; unome$, età
PRINT unome$, età
'Nessun punto di domanda
INPUT "Immettere il nome e l'età", unome$, età
PRINT unome$, età
```

Funzioni correlate: INPUT$

Istruzioni correlate: INPUT #; LINE INPUT

Istruzione INPUT

Descrizione

Legge i dati da un file sequenziale.

Sintassi

INPUT #*numero_file*, *variabile*

Note

- *numero_file* corrisponde al numero assegnato al file nella relativa istruzione OPEN.
- *variabile* corrisponde all'elenco di variabili in cui devono essere registrati i dati dal file.

Esempio

```
OPEN "PAGA.DAT" FOR INPUT AS #1
DO WHILE NOT EOF(1)
INPUT #1, enome$, paga
PRINT enome$, paga
```

```
LOOP
CLOSE #1
```

Funzioni correlate: INPUT$

Istruzioni correlate: INPUT #; LINE INPUT

Funzione INPUT$

Descrizione

Legge il numero specificato di caratteri da un file alla tastiera.

Sintassi

INPUT$(*num_caratteri* [,[#]*numero_file*])

Note

- *num_caratteri* corrisponde al numero di caratteri che INPUT$ deve leggere. Deve essere minore o uguale alla lunghezza del file, che, per default, è 128.
- *numero_file* corrisponde al numero assegnato al file nella relativa istruzione OPEN. Se viene omesso, INPUT$ legge dalla tastiera.

Esempio

```
'Visualizza un file in lettere MAIUSCOLE
OPEN "\CONFIG.SYS" FOR INPUT AS #1
DO WHILE NOT EOF(1)
    car$ = INPUT$(1, 1)
PRINT UCASE$(car$);
LOOP
CLOSE #1
```

Istruzioni correlate: INPUT; INPUT#; LINE INPUT

Funzione INSTR

Descrizione

Restituisce il primo punto in cui si trova la stringa all'interno di un'altra.

Sintassi

INSTR([*inizio*], *stringaricerca*, *sottostringa*)

Note

- INSTR restituisce la posizione del carattere della *sottostringa* all'interno della *stringaricerca*.
- *inizio* corrisponde alla posizione del carattere nella *stringaricerca* dove la ricerca deve iniziare. Se si omette *inizio* INSTR parte dalla posizione 1.
- Se INSTR localizza la *sottostringa*, restituisce un indice al carattere iniziale. Altrimenti, INSTR restituisce 0.

Esempio

```
PRINT "RINGA in SOTTOSTRINGA", INSTR("SOTTOSTRINGA", "RINGA")
PRINT "X in STRINGA", INSTR("STRINGA", "X")
```

Se si lancia questo programma si ottiene quanto segue:

```
RINGA in SOTTOSTRINGA 6
X in STRINGA          0
```

Funzioni correlate: LEFT$; LEN; MID$; RIGHT$

Istruzioni correlate: MID$

Funzione INT

Descrizione

Restituisce il valore intero minore o uguale all'espressione numerica specificata.

Sintassi

INT(*espressione_numerica*)

Note

- *espressione_numerica* corrisponde a una qualsiasi espressione numerica.

Esempio

```
PRINT INT(99.8), INT(99.1), INT(-99.2)
```

Se si lancia questo programma si ottiene:

```
99      99      -100
```

Funzioni correlate: FIX

Istruzione IOCTL

Descrizione

Trasmette una stringa di controllo della periferica al driver di periferica.

Sintassi

IOCTL [#]*numero_file*, *stringa_controllo*

Note

- *numero_file* corrisponde al numero del file assegnato alla periferica nella relativa istruzione OPEN.
- *stringa_controllo* corrisponde alla stringa che specifica il comando da inviare alla periferica.
- Per informazioni riguardo alla stringa di controllo della periferica si faccia riferimento alla documentazione hardware.

Funzioni correlate: IOCTL$

Funzione IOCTL$

Descrizione

Restituisce una stringa da un driver di periferica.

Sintassi

IOCLTL$([#]*numero_file*)

Note

- *numero_file* corrisponde al numero di file assegnato alla periferica nella relativa istruzione OPEN.

- L'informazione che IOCTL$ restituisce è dipendente dalla periferica. Per ulteriori informazioni si consultino i manuali hardware.

Istruzioni correlate: IOCTL

Istruzione KEY

Descrizione

Assegna valori ai tasti funzione da F1 a F12. Se si desidera, può anche visualizzare i valori di ciascun tasto.

Sintassi

KEY *tasto_funzione*, *stringa*
oppure
KEY LIST
oppure
KEY ON
oppure
KEY OFF

Note

- *tasto_funzione* corrisponde al numero del tasto funzione desiderato. 1 corrisponde a F1, 10 a F10. Si utilizzi 30 e 31 per i tasti F11 e F12.
- *stringa* corrisponde a una stringa che può essere composta fino a 15 caratteri che si desidera assegnare al tasto funzione.
- Una volta assegnata una stringa a un tasto, il QBasic inserisce la stringa immessa ogni volta che si preme quel tasto funzione.
- KEY LIST visualizza l'intera stringa di 15 caratteri per ciascun tasto.
- KEY ON visualizza le prime sei lettere della stringa assegnata ai tasti da F1 a F10 sul margine inferiore del video.
- KEY OFF cancella la visualizzazione della stringa assegnata al tasto funzione.

Esempio

```
'Assegna una stringa a F1
KEY 1, "F1 tasto funzione"
'Visualizza la funzione assegnata al tasto
KEY ON
```

```
INPUT "Premere il tasto F1"; x$
PRINT x$
```

Istruzione KEY(*n*)

Descrizione

Attiva o disattiva l'incanalamento del programma per un tasto specificato.

Sintassi

KEY(*n*) **ON**
oppure
KEY(*n*) **OFF**
oppure
KEY(*n*) **STOP**

Note

- *n* corrisponde al numero associato al tasto funzione, a un tasto direzione o a un tasto definito dall'utente.

Valore di *n*	Significato
da 1 a 10	Tasti funzione da F1 a F10
11	↑
12	←
13	→
14	↓
Da 15 a 25	Tasti definiti dall'utente
30 e 31	Tasti funzione F11 e F12

- KEY(*n*) ON attiva l'incanalamento dei dati per un tasto specificato.
- KEY(*n*) OFF disattiva l'incanalamento dei dati per il tasto specificato. Il QBasic non genera una sequenza delle operazioni che si presentano.
- KEY(*n*) STOP blocca l'incanalamento dei dati per il tasto specificato. Le operazioni vengono eseguite quando l'incanalamento viene avviato.
- Dopo aver specificato l'incanalamento delle operazioni da tastiera per un tasto specificato, l'istruzione ON *operazione* GOSUB attiva l'incanalamento da tastiera.

- Per dichiarare un tasto definito dall'utente, si utilizzi la seguente variazione dell'istruzione KEY:

  KEY *n*, **CHR$**(*marcatoretastiera*) + **CHR$**(*codicescan*)

  n corrisponde al numero associato al tasto definito dall'utente (da 15 a 25). *marcatoretastiera* corrisponde a uno dei seguenti valori:

Marcatore	Significato
0	Nessun marcatore di tastiera
da 1 a 3	Consentito anche il tasto Shift*
4	Tasto Ctrl
8	Tasto Alt
32	Tasto Bloc Num
64	Tasto Caps Lock
128	tasti estesi su una tastiera da 101 tasti

** L'incanalamento dei tasti non fa distinzione tra il tasto Shift di destra e quello di sinistra.*

- È possibile aggiungere valori al testo per controllare più marcatori simultanei. *codicascan* corrisponde al codice scan del tasto desiderato.

Esempio

```
ON KEY(10) GOSUB Handler
KEY(10) ON
PRINT "Premere F10 per interrompere"
FOR i = 0 TO 100000
    PRINT i
NEXT i
Handler:
STOP
```

Istruzioni correlate: ON *operazione* GOSUB

Istruzione KILL

Descrizione

Cancella un file dal disco.

Sintassi

KILL *file*

Note

- *file* corrisponde a una stringa indicante il file che deve essere cancellato. All'interno della stringa è consentito l'uso di caratteri jolly (* e ?).

Esempio

```
KILL "TEST.DAT"
KILL "*.VEC"
```

Istruzioni correlate: FILES

Funzione LBOUND

Descrizione

Restituisce il più basso indice inferiore dell'array per le dimensioni dell'array specificato.

Sintassi

LBOUND(*nome_array*[,*dimensioni*])

Note

- *nome_array* corrisponde al nome dell'array desiderato.
- *dimensioni* corrisponde al valore intero che specifica la dimensione desiderata in un array multidimensionale. Il valore di default è 1.

Esempio

```
DIM a(50 TO 100) AS INTEGER
DIM box(1 TO 3, 3 TO 6) AS INTEGER
PRINT LBOUND(a)
PRINT LBOUND(box, 1), LBOUND(box, 2)
```

Se si lancia questo programma, viene visualizzato quanto segue:

```
50
1    3
```

Funzioni correlate: UBOUND

Istruzioni correlate:DIM

Funzione LCASE$

Descrizione

Restituisce una stringa di caratteri con tutte le lettere nella stringa specificata in caratteri minuscoli.

Sintassi

LCASE$(*stringa*)

Note

- *stringa* corrisponde a una stringa qualsiasi.

Esempio

```
INPUT "Immettere una stringa"; S$
PRINT LCASE$(S$)
```

Funzioni correlate: UCASE$

Funzione LEFT$

Descrizione

Restituisce il numero di caratteri specificato partendo dal primo carattere a sinistra della stringa.

Sintassi

LEFT$(*stringa*, *num_carattere*)

Note

- *stringa* corrisponde a una stringa qualsiasi.
- *num_carattere* corrisponde al numero di caratteri da estrarre dalla stringa e deve essere compreso nell'intervallo tra 0 e 32.767.

Esempio

```
s$ = "STRINGA TEST"
FOR i = 1 TO LEN(S$)
    PRINT LEFT$(s$, I)
NEXT i
```

Funzioni correlate: INSTR; LEN; MID$; RGHT$

Istruzioni correlate: MID$

Funzione LEN

Descrizione

Restituisce il numero di caratteri in una stringa o il numero di byte utilizzati per memorizzare una variabile.

Sintassi

LEN(stringa)
oppure
LEN (*variabile*)

Note

- *stringa* corrisponde a una stringa qualsiasi.
- *variabile* è una qualsiasi variabile di un tipo diverso da STRING.

Esempio

```
DIM x AS INTEGER, y AS LONG
a$ = "13 CARATTERI"
PRINT s$, LEN(a$)
PRINT "Integer"; LEN(x), "Long"; LEN(y)
```

Se si lancia questo programma, viene visualizzato quanto segue:

```
13 CARATTERI 13
Integer 2   Long 4
```

Funzioni correlate: INSTR; LEFT$; MID$; RIGHT$

Istruzioni correlate: MID$

Istruzione LET

Descrizione

Assegna un valore a una variabile.

Sintassi

[**LET**] *variabile = espressione*

Note

- LET è una parola chiave facoltativa utilizzata per assegnare un valore a una variabile.

Esempio

```
LET a = 5
'assegnazione equivalente senza LET
a = 5
```

Istruzione LINE

Descrizione

Disegna una riga o un quadrato.

Sintassi

LINE[[**STEP**](*x1*, *y1*)]-[**STEP**](*x2*,*y2*)[,[*colore*][,[B[F]][[*stileriga*]]]

Note

- Questa istruzione consente di disegnare una riga utilizzando le coordinate *x1* e *y1* e *x2* e *y2* per indicare il punto in cui la riga inizierà e quello in cui finirà; se invece si desidera disegnare un quadrato si utilizzino *x1 y1* e *x2* e *y2* come coordinate per i due angoli della figura.
- Indica che le coordinate sono relative alla posizione corrente del cursore grafico.
- *color* corrisponde al colore del quadrato della riga.
- B indica a LINE di disegnare un quadrato anziché una riga.
- F indica a LINE di riempire il quadrato con un colore specificato.
- *stileriga* corrisponde a un valore di 116 byte i cui bit determinano se i pixel vengono disegnati o meno.

Esempio

```
'Riempie il video con quadrati posizionati in modo casuale.
SCREEN 1
```

```
FOR i = 1 TO 1000
x1 = RND * 320
y1 = RND * 200
x2 = RND * 320
y2 = RND * 200
SCOLORE = RND * 4
LINE (x1, y1)-(x2, y2), scolor, BF
NEXT i
```

Funzioni correlate: POINT

Istruzioni correlate: CIRCLE; COLOR; DRAW; PAINT; PRESET; PSET; SCREEN

Istruzione LINE INPUT

Descrizione

Legge fino a 255 caratteri in una stringa

Sintassi

LINE INPUT [;]["*prompt*";]*variabile_stringa*
oppure
LINE INPUT [#] *numero_file*, *stringa_variabile*

Note

- A differenza dell'istruzione INPUT, LINE INPUT non interpreta la virgola come un separatore tra due voci. Questa istruzione legge tutti i caratteri fino al ritorno di carrello e li assegna alla stringa variabile.
- Se si inserisce un punto e virgola subito dopo la parola chiave INPUT; questo indica a LINE INPUT di lasciare il cursore nella stessa riga dopo che si è premuto Invio.
- *prompt* corrisponde a un messaggio facoltativo che indica all'utente di immettere la data.
- *variabile_stringa* corrisponde alla variabile stringa a cui LINE INPUT assegna le informazioni immesse.
- *numero_file* corrisponde al numero di file assegnato al file nella relativa istruzione OPEN

Esempio

```
LINE INPUT "Immettere nome, cognome, MI: "; nomecompleto$
PRINT nomecompleto$
```

Funzioni correlate: INPUT$

Istruzioni correlate: INPUT; INPUT#

Funzione LOC

Descrizione

Restituisce l'offset corrente o il numero del record all'interno di un file.

Sintassi

LOC(*numero_file*)

Note

- *numero_file* corrisponde al numero del file assegnato al file stesso nell'istruzione OPEN.
- Per i file binari, LOC restituisce il byte di offset corrente del file. Per i file ad accesso casuale, LOC restituisce il numero del record corrente; per i file sequenziali, LOC restituisce l'offset corrente, diviso per 128. Per una periferica COM, LOC restituisce il numero dei byte della coda di input.

Esempio

```
IF LOC(1) > 100 THEN CALL Leggi(paghe)
```

Funzioni correlate: SEEK

Istruzioni correlate: SEEK

Istruzione LOCATE

Descrizione

Sposta il cursore nella posizione specificata sul video e, se lo si desidera, imposta la dimensione del cursore.

Sintassi

LOCATE [*riga*][,[*colonna*][,[*visibile*][,[*inizio_scan*][,[*fine_scan*]]]]

Note

- *riga* corrisponde al numero di riga desiderata.
- *colonna* corrisponde al numero di colonna desiderata.
- quando *visibile* ha valore 1, il cursore viene visualizzato, quando il valore corrisponde a 0, il cursore rimane nascosto.
- *inizio_scan*, *fine_scan* corrispondono a valori interi compresi tra 0 e 31 che indicano la prima e l'ultima riga di espansione del cursore.
- È possibile modificare la dimensione del cursore modificandone le righe di espansione.

Esempio

```
CLS
FOR i = 5 TO 20
     LOCATE i, i
PRINT "La posizione è"; i; i
NEXT i
```

Funzioni correlate: CSRLIN; POS

Istruzione LOCK

Descrizione

Limita o impedisce l'accesso a un file condiviso in una rete.

Sintassi

LOCK [#]*numero_file* [,{*record* | [*inizio*] **TO** *end*}]

Note

- *numero_file* corrisponde al numero assegnato al file nella relativa istruzione OPEN.
- Per l'accesso casuale ai file, LOCK blocca l'accesso ai record specificati o alla serie di record. Per file binari, LOCK chiude il byte specificato o l'intervallo di byte. Per i file in sequenza, chiude l'intero file.

- LOCK è necessario solo in un ambiente a rete.
- Se un programma in rete tenta di accedere a un record o a un byte bloccato, QBasic visualizza un messaggio d'errore.
- Per utilizzare LOCK, è prima necessario lanciare il programma SHARE.EXE, un programma MS-DOS che gestisce la condivisione e il blocco dei file.

Esempio

```
INPUT "Immettere numero di record da aggiornare"; rec
LOCK #1, rec        'Accesso limitato
emp.nome$ = "SMITH"
PUT #1, rec
UNLOCK rec          'Consente accesso
```

Istruzioni correlate: UNLOCK

Funzione LOF

Descrizione

Restituisce il numero di byte all'interno di un file.

Sintassi

LOF(*numero_file*)

Note

- *numero_file* corrisponde al numero assegnato al file nella relativa istruzione OPEN.
- Non è consentito l'uso della funzione LOF con periferiche.

Esempio

```
OPEN "\CONFIG.SYS" FOR INPUT AS #1
PRINT "La dimensione del file in byte è"; LOF(1)
CLOSE #1
```

Funzione LOG

Descrizione

Restituisce il logaritmo naturale di un'espressione numerica.

Sintassi

LOG(*espressione_numerica*)

Note

- *espressione_numerica* corrisponde a un'espressione numerica qualsiasi.
- Il logaritmo naturale è il logaritmo in base *e*.

Esempio

```
PRINT LOG(1), LOG(EXP(1))
```

Se si lancia questo programma, viene visualizzato quanto segue:

```
0             1
```

Funzione correlata: EXP

Funzione LPOS

Descrizione

Restituisce alla posizione corrente la testina della stampante di un buffer di stampa.

Sintassi

LPOS(*numero_stampante*)

Note

- *numero-stampante* corrisponde al numero della stampante desiderata. 1 è uguale a LPT1, 2 a LPT2 ecc.
- Non tutte le stampanti riconoscono la funzione LPOS.

Esempio

```
FOR i = 1 TO 100
     LPRINT i;          'Stampa il numero sulla stessa riga
     IF LPOS(1) > 50 THEN LPRINT  'Inizia una nuova riga
NEXT i
```

Istruzione LPRINT

Descrizione

Stampa sulla stampante LPT1.

Sintassi

LPRINT[*elenco_output*][{; | ,}]

Note

- *elenco_output* corrisponde a un elenco di numeri e a una stringa che deve essere stampata. Le espressioni devono essere separate con virgole o punti e virgola.
- Se si conclude la lista di output con un punto e virgola, la testina della stampante rimane sul carattere successivo. Se si conclude con una virgola, la testina viene portata nella zona di stampa successiva (la lunghezza delle zone di stampa corrisponde a 14 caratteri). Se si omette sia la virgola sia il punto e virgola, la stampante avanza fino all'inizio della riga successiva.

Esempio

```
LPRINT "Questo si trova sulla riga 1"
LPRINT "Questo si trova";
LPRINT "riga 2"
```

Se si lancia questo programma, viene visualizzato quanto segue:

```
Questo si trova sulla riga 1
Questo si trova sulla riga 2
```

Istruzioni correlate: LPRINT USING; WIDTH

Istruzione LPRINT USING

Descrizione

Stampa output formattati sulla stampante LPT1.

Sintassi

LPRINT USING *stringa_formato*; *elenco_output*[{; | ,}]

Note

- *stringa_formato* corrisponde al formato dell'output. Si veda PRINT USING per un elenco dei caratteri di formattazione.
- *elenco_output* corrisponde a un elenco di stringhe e di espressioni numeriche che devono essere stampate. Le espressioni devono essere separate da virgole o punti e virgola.
- Se si conclude la lista di output con un punto e virgola, la testina della stampante rimane sul carattere successivo. Se si conclude con una virgola, la testina resta nella zona di stampa successiva (la lunghezza delle zone di stampa corrisponde a 14 caratteri). Se si omette sia la virgola sia il punto e virgola, la stampante avanza fino all'inizio della riga successiva.

Esempio

Si veda PRINT USING.

Istruzioni correlate: LPRINT; PRINT USING; WIDTH

Istruzione LSET

Descrizione

Sposta i dati in un buffer di file ad accesso casuale, assegna una variabile di un tipo di record a una variabile di un tipo di un record differente, o giustifica a sinistra il valore di una variabile stringa.

Sintassi

LSET *variabile_stringa* = *espressione_stringa*
oppure
LSET *variabile1_record* = *variabile2_record*

Note

- *variabile_stringa* corrisponde sia a una campo file ad accesso casuale sia a una variabile stringa.
- *espressione_stringa* corrisponde a una stringa qualsiasi.
- *variabile1_record* e *variabile2_record* sono variabili record definite dall'utente.

Esempio

```
stipendi = 1500000
LSET e$ = MKS$(stipendio)
PUT #1
```

Istruzioni correlate: RSET

Funzione LTRIM$

Descrizione

Cancella i caratteri di riempimento da una stringa.

Sintassi

LTRIM$(*espressione_stringa*)

Note

- *stringa* corrisponde a una stringa qualsiasi.

Esempio

```
PRINT LTRIM$("      Trim test")
```

Se si lancia questo programma, viene visualizzato quanto segue:

```
Trim test
```

Funzioni correlate: RTRIM$

Funzione MID$

Descrizione

Restituisce una sotto stringa che inizia all'offset indicato.

Sintassi

MID$(*stringa*, *offset_iniziale* [,*lunghezza*])

Note

- *stringa* corrisponde a una stringa qualsiasi.
- *inizio_offset* corrisponde alla posizione del primo carattere nella sottostringa.
- *lunghezza* corrisponde al numero di caratteri nella sottostringa. Se si omette *lunghezza* MID$ restituisce tutti i caratteri da *inizio_offset* fino alla fine della stringa.

Esempio

```
a$ = "ABCDEFGHI"
PRINT  MID$(a$, 1, 5)
PRINT  MID$(a$, 6)
```

Se si lancia questo programma, viene visualizzato quanto segue:

```
ABCDE
FGHI
```

Funzioni correlate: INSTR; LEFT$; LEN; RIGHT$

Istruzioni correlate: MID$

Istruzione MID$

Descrizione

Sostituisce una sezione di una stringa con un'altra.

Sintassi

MID$(*variabile_stringa*, *inizio_offset*[,*num_car*]) = *espressione_stringa*

Note

- *variabile_stringa* corrisponde alla stringa che deve essere modificata.
- *inizio_offset* corrisponde alla posizione del primo carattere nella stringa che deve essere sostituita. Se si omette questo valore, MID$ utilizza la lunghezza della stringa che verrà sostituita.
- *num_car* corrisponde al numero dei caratteri della stringa che dovranno essere sostituiti. Se questo valore viene omesso, MID$ utilizza la lunghezza della stringa sostitutiva.
- *espressione_stringa* corrisponde a una stringa qualsiasi.

Esempio

```
a$ = ABCDEF"
 MID$(a$, 2, 2) = "bc"
PRINT a$
```

Se si lancia questo programma, viene prodotto quanto segue:

```
AbcDEF
```

Funzioni correlate: INSTR; LEFT$; LEN; MID$; RIGHT$

Funzione MKD$

Descrizione

Converte un valore a precisione doppia in una stringa a 8-byte per output a un file ad accesso casuale avviato tramite PUT.

Sintassi

MKD$(*espressione_numerica*)

Note

- *espressione_numerica* corrisponde a un'espressione numerica a doppia precisione.

Esempio

```
OPEN "TEST.DAT" FOR RANDOM AS #1 LEN=8
FIELD #1, 8 AS sti$
stipendi = 6000
sti$ = MKD$(stipendi)
```

```
PUT #1
CLOSE #1
```

Funzioni correlate: CVD; CVI; CVL; CVS; CVDMBF; CVSMBF; MKDMBF$; MKI$; MKL$; MKS$; MKSMBF$

Istruzione MKDIR

Descrizione

Crea la sottodirectory MS-DOS specificata.

Sintassi

MKDIR *nome_directory*

Note

- *nome_directory* corrisponde alla stringa che indica la sottodirectory da creare.

Esempio

```
ON ERROR GOTO CheckExist
MKDIR "TEST.DIR"
PRINT "Directory TESTDIR creata"
Eseguito:
END
CheckExist:
       IF ERR = 75 THEN
          PRINT "Directory TESTDIR esiste già"
          RESUME Eseguito
        END IF
        ON ERROR GOTO 0
```

Istruzioni correlate: CHDIR; RMDIR

Funzione MKDMBF$

Descrizione

Converte un valore a doppia precisione registrato in formato IEEE in una stringa a 8 byte contenente il valore in formato binario Microsoft per output a un file ad accesso casuale di PUT.

Sintassi

MKDMBF$(*espressione_numerica*)

Note

- *espressione_numerica* corrisponde a un'espressione numerica a precisione doppia.
- CVDMBF, CVSMBF, MKDMBF$, MKSMBF$ sono utili per mantenere i file di dati creati con le versioni precedenti di Basic.

Esempio

Si veda MKSMBF$

Funzioni correlate: CVD, CVDMBF; CVI; CVL, CVS; CVSMBF; MKD$; MKI$; MKL$; MKS$; MKSMBF$

Funzione MKI$

Descrizione

Converte un valore intero a una stringa di 2 byte per un output a un file ad accesso casuale tramite PUT.

Sintassi

MKI$(*espressione_numerica*)

Note

- *espressione_numerica* corrisponde a un'espressione numerica intera.

Esempio

Si veda MKD$

Funzioni correlate: CVD; CVDMBF$; CVI; CVL; CVS; CVSMBF$; MKD$; MKDMBF$; MKL$; MKS$ MKSMBF$

Funzione MKL$

Descrizione

Converte un valore intero di tipo long in una stringa a 4 byte per output a un file ad accesso casuale tramite PUT.

Sintassi

MKL$(*espressione_numerica*)

Note

- *espressione_numerica* corrisponde a un'espressione numerica intera di tipo LONG.

Esempio

Si veda MKD$

Funzioni correlate: CVD; CVDMBF; CVI; CVL; CVS; CVSMBF; MKD$; MKDMBF$; MKI$; MKS$; MKSMBF$

Funzione MKS$

Descrizione

Converte un valore a precisione singola in una stringa a 4 byte per un file ad accesso casuale tramite PUT.

Sintassi

MKS$(*espressione_numerica*)

Note

- *espressione numerica* corrisponde a un'espressione numerica a precisione singola.

Esempio

Si veda MKD$

Funzioni correlate: CVD; CVDMBF; CVI; CVL; CVS; CVSMBF; MKD$; MKDMBF$; MKI$; MKL$; MKSMBF$

Funzione MKSMBF$

Descrizione

Converte un valore a precisione singola registrato in formato IEEE in una stringa a 4 byte contenente il valore in formato binario Microsoft per output in un file ad accesso casuale tramite PUT.

Sintassi

MKSMBF$(*espressione_numerica*)

Note

- *espressione_numerica* corrisponde a un'espressione numerica a precisione singola.
- CVDMBF, CVSMBF, MKDMBF$ e MKSMBF$ sono utili per mantenere i file di dati creati con versioni precedenti di Basic.

Esempio

```
TYPE DipRec
   dnome AS STRING * 20
   stipendi AS STRING * 4
END TYPE

DIM dipendenti AS Diprec
OPEN "STIPENDI.DAT" FOR RANDOM AS #1 LEN=LEN(dipendente)
dipendente.dnome = "Rossi"
dipendente.stipendi = MKSMBF$(1450000)
PUT #1, 1, dipendente
CLOSE #1
```

Funzioni correlate: CVDMBF; CVSMBF; MKDMBF$

Istruzione NAME

Descrizione

Assegna un nuovo nome a una directory o a un disco.

Sintassi

NAME *nome_filevecchio* **AS** *nome_filenuovo*

Note

- *nome_vecchiofile* corrisponde alla stringa contenente il nome di un file MS-DOS già esistente.
- *nome_nuovofile* corrisponde alla stringa del nuovo nome che si assegna al file, che non deve esistere già tra i file MS-DOS.
- *nome_nuovofile* deve trovarsi sullo stesso disco di *nome_vecchiofile*.

Esempio

```
NAME "VECCHIO.DAT" AS "NUOVO.DAT"
```

Istruzioni correlate: FILES

Funzione OCT$

Descrizione

Restituisce una stringa contenente la rappresentazione ottale di un'espressione.

Sintassi

OCT$(*espressione_numerica*)

Note

- Ottale è la base 8 del sistema di numerazione.
- *espressione_numerica* corrisponde a un'espressione numerica qualsiasi.

Esempio

Si veda HEX$

Funzioni correlate: HEX$

Istruzione ON ERROR GOTO

Descrizione

Attiva la gestione degli errori e specifica la prima riga del gestore di errore.

Sintassi

ON ERROR GOTO *locazione*

Note

- Vedi ERR per un elenco degli errori possibili.
- *locazione* corrisponde a numero di riga o all'etichetta della prima riga nel gestore. Il numero di riga 0 disattiva la gestione dell'errore.
- Se la gestione dell'errore viene disattivata, in caso di errore viene visualizzato un messaggio di errore e il programma viene interrotto.

Esempio

```
ON ERROR GOTO Handler
OPEN "NOFILE.DAT" FOR INPUT AS #1
PRINT "File aperto"
CLOSE #1
Eseguito:
END
Handler:
    IF ERR = 53 THEN
         PRINT "File non trovato"
    END IF
RESUME Eseguito
```

Funzioni correlate: ERDEV; ERDEV$; ERL; ERR

Istruzioni correlate: ERROR; RESUME

Istruzione ON operazione GOSUB

Descrizione

Specifica la prima riga di una subroutine di incanalamento.

Sintassi

ON COM(*n*) **GOSUB** *locazione*
oppure
ON KEY(*n*) **GOSUB** *locazione*
oppure
ON PEN GOSUB *locazione*
oppure
ONPLAY(*dimensioni_coda*) **GOSUB** *locazione*
oppure
ON STRIG(*n*) **GOSUB** *locazione*
oppure
ON TIMER(*n*) **GOSUB** *locazione*

Note

- L'istruzione **ON COM**(*n*) **GOSUB** *locazione* salta alla subroutine quando i caratteri vengono ricevuti alla porta seriale specificata.
- **ON KEY**(*n*) **GOSUB** *locazione* salta alla subroutine quando viene premuto il tasto associato al numero specificato.
- **ON PEN GOSUB** *locazione* salta alla subroutine quando è attivata la penna ottica.
- **ONPLAY**(*dimensioni_coda*) **GOSUB** *locazione* salta alla subroutine quando il numero di note nel buffer musicale va da *dimensioni_coda* a *dimensioni_coda* – 1. La variabile *dimensioni_coda* può essere compresa nell'intervallo tra 1 e 32.
- **ON STRIG**(*n*) **GOSUB** *locazione* salta alla subroutine quando viene premuto il pulsante specificato del joystick. Si veda l'istruzione STRIG per i valori validi di *n*.
- **ON TIMER**(*n*) **GOSUB** *locazione* salta alla subroutine quando è trascorso il numero specificato di secondi, che deve essere compreso nell'intervallo tra 1 e 86.400.

- Queste istruzioni non attivano le operazioni di incanalamento; esse associano semplicemente una subroutine a un'operazione.

Esempio

```
ON KEY(10) GOSUB VisualizzaGuida
KEY(10) ON

PRINT "Premere F10 per la guida"
DO
    INPUT "Immettere il nome"; n$
    PRINT n$
LOOP UNTIL n$ = "ESCI"
END

VisualizzaGuida:
    PRINT "Digitare ESCI per uscire"
    RETURN
```

Istruzioni correlate: COM; KEY(n); PEN; PLAY (Incanalamento); STRIG; TIMER

Istruzione ON *espressione*

Descrizione

Salta a una delle numerose locazioni in base al risultato di un'espressione.

Sintassi

ON *espressione_numerica* **GOSUB** *posizione*[,*posizione*]...
oppure
ON *espressione_numerica* **GOTO** *posizione*[, *posizione*]...

Note

- *espressione_numerica* corrisponde a un'espressione numerica compresa tra 0 e 255. Se necessario, QBasic arrotonda l'espressione a una valore intero.

- *posizione* corrisponde all'elenco di numeri di riga a cui deve saltare il controllo. Se il risultato dell'espressione numerica è 1, il controllo salta alla prima posizione. Se il risultato è 2, il controllo salta alla seconda e così via. Se il valore non corrisponde a nessuna etichetta l'esecuzione prosegue all'istruzione successiva.

Esempio

```
FOR I = 1 TO 3
      ON i GOSUB Uno, Due, Tre
NEXT i
END
Uno:
      PRINT "In Uno"
RETURN
Due:
      PRINT "In Due"
RETURN
Tre:
      PRINT "In Tre"
RETURN
```

Istruzioni correlate: DO UNTIL; DO WHILE; GOSUB; GOTO

Istruzione OPEN

Descrizione

Apre un file o una periferica per operazioni di input o output.

Sintassi

OPEN *nomefile* [**FOR** *modo_accesso*][**ACCESS** *accesso_rete*][*tipo_lock*]**AS** [#]*numero_file*[**LEN**=*lunghezza_record*]
oppure
OPEN *modo*, [#]*numero_file*[,*lunghezza_record*]
oppure
OPEN "**COM***n*: *com_basic com_specifici*"[**FOR** *modo_accesso*]**AS** [#]*numerofile*[**LEN**=*lunghezza_record*]

Note

- *nomefile* corrisponde a una stringa contenente il nome del file o della periferica da aprire.
- *modo_accesso* specifica come il file deve essere utilizzato: INPUT, OUTPUT, APPEND, RANDOM, o BINARY. La modalità di default è RANDOM.
- *accesso_rete* fornisce più dettagli su come deve essere utilizzato un file in ambienti rete a condivisione di file: READ, WRITE oppure READ e WRITE.

- *tipo_lock* corrisponde al tipo blocco di file usato in ambienti a condivisione di file: SHARED, LOCK READ, LOCK WRITE, oppure LOCK READ WRITE.
- *numero_file* corrisponde al numero intero che viene associato al file per la lettura, scrittura e per le operazioni di tipo close.
- *lunghezza_record* corrisponde al numero di byte in ciascun record. Per i file in sequenza, il valore di default è 512; per i file ad accesso casuale è 128; per le comunicazioni, 128. Il valore non può comunque superare 32.767.
- *modo* viene utilizzato per i più vecchi programmi Basic. È una stringa composta da una sola lettera che specifica il modo di accesso:

Opzione	**Modo**
A	Append
B	Binary
I	Input
O	Output
R	Random

- *n* corrisponde al numero di porta di comunicazione da aprire, sia la 1 che la 2.
- *basic_com* corrisponde ai parametri di comunicazione dati in basic, che vengono separati da virgole, nel seguente modo:
  [*baud*][,[*parità*][*bit_dati*][,[*bit_stop_dati*]]]]
  - *baud* corrisponde al baud rate della periferica utilizzata: 75, 110, 150, 300, 600, 1200, 1800, 2400, 4800, 9600 o 19200.
  - *parità* corrisponde alla parità della periferica utilizzata: N (nessuna), E (pari), O (dispari), S (spazio), M (annotazione) o PE (attiva controllo errori).
  - *bit_dati* corrisponde al numero di bit in ciascuna parola di dati: 5, 6, 7 o 8. La somma dei bit di parità e dei bit di dati deve essere minore o uguale a 8. Se la parità è impostata su O oppure su E, viene utilizzato un bit per la parità. Se la parità è impostata su N non vengono utilizzati bit per la parità.
  - *bit_stop_dati* corrisponde al numero di bit di stop per ciascuna parola: 1, 1.5 oppure 2.
- *com_specifico* è un elenco di comunicazioni di dati specifici separati tra loro da virgole:

Opzione	Funzione
ASC	Apre una periferica in modalità ASCII.
BIN	Apre una periferica in modalità binaria.
CD[millesimi di secondo]	Specifica il tempo di time-out sulla riga Data Carrier Detect (DCD).
CS[millesimi di secondo]	Specifica il tempo di time-out sulla riga Pronto per Trasmettere (CTS).
DS[millesimi di secondo]	Specifica il tempo in time-out sulla riga Insieme di Dati Pronto (DS).
LF	Carattere nuova riga dopo un ritorno di carrello.
OP[millesimi di secondo]	Indica l'intervallo di tempo che l'istruzione OPEN deve attendere prima che si aprano tutte le righe di comunicazione.
RB[byte]	Specifica le dimensioni in byte del buffer di ricezione.
RS	Disattiva la modalità di individuazione della Richiesta di Emissione (RST).
TB[byte]	Specifica le dimensioni del buffer di trasmissione.

Esempio

```
OPEN "TEST.DAT" FOR INPUT AS #1

f% = FREEFILE
OPEN "NUOVO.DAT" FOR RANDOM AS f% LEN = 80

OPEN "COM1:4800, E, 7, 1, BIN" AS #2
```

Funzioni correlate: FREEFILE

Istruzioni correlate: CLOSE

Istruzione OPTION BASE

Descrizione

Imposta il limite inferiore predefinito per gli indici inferiori di array.

Sintassi

OPTION BASE {0 | 1}

Note

- Se il limite inferiore non viene specificato, il QBasic utilizza 0.
- Può essere specificata solo un'istruzione OPTION BASE per modulo.
- Per maggiore flessibilità, si utilizzi la condizione TO nell'istruzione DIM.

Esempio

```
OPTION BASE 0
DIM a(1 TO 20)
DIM b(20)
PRINT LBOUND(a), LBOUND(b)
PRINT UBOUND(a), UBOUND(b)
```

Se si lancia questo programma, viene visualizzato quanto segue:

```
1      0
20      20
```

Funzioni correlate: LBOUND; UBOUND

Istruzioni correlate: DIM; REDI

Istruzione OUT

Descrizione

Invia un byte a una porta specificata.

Sintassi

OUT *numeri_porta*, *valore_byte*

Note

- *numero_porta* corrisponde a un'espressione intera nell'intervallo tra 0 e 65.535 che identifica la porta.
- *valore_byte* corrisponde a un'espressione intera nell'intervallo tra 0 e 255 da inviare alla porta.

Esempio

Si veda INP

Funzione correlata: INP

Istruzione PAINT

Descrizione

Riempie un'immagine grafica con il colore o il modello specificato.

Sintassi

PAINT [**STEP**](*x,y*)[,[{*colore*|*casella*}][,[*colore_bordo*][,*sfondo*]]]

Note

- La parola chiave STEP indica che le coordinate *x* e *y* sono relative alla posizione corrente del cursore grafico.
- *x,y* corrispondono a una serie di coordinate dello schermo in cui inizia il disegno.
- *colore* corrisponde a un attributo di colore; se viene omesso, PAINT utilizza il colore principale corrente.
- *casella* corrisponde a un modello di riempimento largo 8 bit e lungo fino a 64 byte che viene specificato come segue:
  *casella$*=CHR$(n)[+CHR$(n)]...
  dove i valori di *n* sono gli argomenti utilizzati con CHR$ con valori compresi tra 0 e 255. Ciascuna funzione CHR$(*n*) definisce una striscia da 1 byte di 8 pixel del disegno in base alla forma binaria del numero.
- *colore_bordo* corrisponde al colore del bordo dell'immagine grafica; se viene omesso PAINT utilizza *colore*.
- *sfondo* corrisponde a una striscia di casella di sfondo da 1 byte di 8 pixel con cui è possibile riempire un'area che era stata riempita in precedenza. Se si omette *colore_sfondo* PAINT utilizza CHR$(0).

Esempio

```
SCREEN 1
LINE (10, 10)-(50, 50), 1, B
PAINT (11, 11), 2, 1
```

Funzione correlata: POINT

Istruzioni correlate: CIRCLE; COLOR; DRAW; LINE; PRESET; PSET; SCREEN

Istruzioni PALETTE, PALETTE USING

Descrizione

Modifica uno o più colori della tavolozza.

Sintassi

PALETTE [*cambia-colore*, *nuovo_colore*]
oppure
PALETTE USING *array*[(*indice*)]

Note

- *cambia_colore* corrisponde all'attributo da cambiare.
- *nuovo_colore* corrisponde al nuovo colore da assegnare all'attributo.
- *array* corrisponde a un array di numeri associati al colore da assegnare agli attributi disponibili nella modalità video attiva.
- *indice* corrisponde all'indice del primo elemento nell'array da utilizzare nell'impostazione della palette.
- Se si omettono tutti gli argomenti, PALETTE ripristina i valori per i colori di default.

Esempio

```
PALETTE 0, 1
```

Istruzioni correlate: COLOR; SCREEN

Istruzione PCOPY

Descrizione

Copia una pagina visualizzata a video su un'altra.

Sintassi

PCOPY *pagina_sorgente*, *pagina_destinazione*

Note

- *pagina_sorgente* corrisponde a un'espressione intera che specifica la pagina a video che deve essere copiata.
- *pagina_destinazione* corrisponde a un'espressione intera che specifica la pagina a video su cui deve essere copiata la pagina sorgente.
- Il numero di pagine video disponibili dipende dalla capacità della memoria video e dalla modalità video.

Esempio

```
'Copia la pag 1 sulla pag 3
PCOPY 1, 3
```

Funzione PEEK

Descrizione

Restituisce un valore di byte contenuto in una locazione di memoria specificata.

Sintassi

PEEK(*offset*)

Note

- *offset* corrisponde a un'espressione intera (nell'intervallo tra 0 e 65.535) che specifica un offset all'interno del segmento di default attivo.
- L'istruzione DEF SEG definisce l'indirizzo del segmento per default.

Esempio

```
'Registra il contenuto dello schermo
DIM salvavideo(3999) AS INTEGER
DEF SEG = &HB800
FOR i = 0 TO 3999
        salvavideo(i) = PEEK(i)
NEXT i
DEF SEG
```

Istruzioni correlate: DEF; POKE

Funzione PEN

Descrizione

Restituisce le coordinate della penna ottica.

Sintassi

PEN(*espressione_numerica*)

Note

- *espressione_numerica* corrisponde a una valore intero (nell'intervallo tra 0 e 9) che specifica l'informazione che PEN restituisce:

Valore	Riporto
0	–1 se la penna è stata utilizzata dall'ultima chiamata; altrimenti 0.
1	Le coordinate *x* pixel nel punto in cui la penna è stata premuta l'ultima volta.
2	Le coordinate *y* pixel nel punto in cui la penna è stata premuta l'ultima volta.
3	Utilizzo corrente: –1 se in basso; 0 se in alto.
4	L'ultimo valore conosciuto di *x* pixel.
5	L'ultimo valore conosciuto di *y* pixel.
6	Riga del carattere in cui la penna è stata premuta.
7	Colonna del carattere in cui la penna è stata premuta.
8	L'ultima riga conosciuta del carattere.
9	L'ultima colonna conosciuta del carattere.

- PEN non funziona quando è attivo il driver del mouse.

Esempio

```
row% = PEN(6)
column = PEN(7)
```

Istruzione PEN

Descrizione

Attiva o disattiva l'incanalamento dei dati con la penna ottica.

Sintassi

PEN ON
oppure
PEN OFF
oppure
PEN STOP

Note

- L'istruzione PEN ON attiva l'incanalamento dei dati tramite penna ottica.
- L'istruzione PEN OFF disattiva l'incanalamento dei dati tramite penna ottica. Tutte le operazioni con penna ottica vengono ignorate.
- L'istruzione PEN STOP sospende temporaneamente le operazioni di incanalamento. Tutte le operazioni eseguite durante questo intervallo vengono eseguite dopo la riattivazione delle operazioni di incanalamento.

Esempio

```
ON PEN GOSUB Handler
PEN ON
```

Funzioni correlate: PEN

Istruzioni correlate: ON operazione GOSUB

Funzione PLAY

Descrizione

Restituisce il numero di note nella coda musicale di sfondo.

Sintassi

PLAY(*argomento_muto*)

Note

- *argomento_muto* corrisponde a un argomento numerico qualsiasi. Distingue semplicemente la funzione Play dall'istruzione omonima.
- PLAY restituisce 0 quando la musica è in modalità principale.

Esempio

```
notes% = PLAY(0)
```

Istruzioni correlate: ON operazione GOSUB; PLAY; PLAY (operazione di incanalamento); SOUND

Istruzione PLAY

Descrizione

Suona quanto specificato nella stringa.

Sintassi

PLAY *stringa*

Note

- *stringa* corrisponde a una stringa contenente uno o più dei seguenti comandi:

Comandi per ottave

>	Sale da un minimo di 1 ottava a un massimo di 6.
<	Scende da un'ottava a un minimo di 0.
O *livello*	Imposta l'ottava corrente; il valore di *livello* può variare da un minimo o 0 a un massimo di 6. Il valore di default è 4.

Comandi di durata

L *nota*	Imposta la lunghezza di ciascuna nota da 1 a 64; 1 corrisponde alla nota intera, 2 a mezza nota, e così via. Il valore impostato per default è 4.
ML	Imposta il legato.
MN	Imposta l'esecuzione normale delle note (impostato per default).
MS	Imposta lo staccato

Comandi modalità

MF Imposta musica in primo piano (impostato per default).

MB Imposta la musica in sottofondo. Può suonare fino a 32 note durante l'esecuzione del programma.

Comandi di tempo

P *durata* Specifica una pausa da 1 a 64; 1 corrisponde a una pausa di semibreve, 2 a una pausa minima ecc.

T *note* Imposta il tempo in quarti di note per minuto e può variare da 32 a 255. Il valore impostato per default è 120.

Comandi di tono

A-G Suona la nota specificata nell'ottava corrente.

N *nota* Suona una nota da 0 a 84 compresa tra sette ottave (0 corrisponde a una pausa).

Comandi di suffisso

o + Emette una nota acuta.

– Emette una nota bassa.

•* Emette una nota di 3/2 per l'intervallo di tempo specificato.

* *Per motivi tipografici è stato inserito il simbolo • che indica un semplice punto (.).*

- La stringa "X"+VARPTR$(*stringa*) esegue la sottostringa musicale *string*.

Esempio

```
'Suona le scale in 7 differenti ottave.
scale$ = "CDEFGAB"
PLAY "L!&"
FOR i = 0 TO 6
      PLAY "O" + STR$(i)
      PLAY "X" * VAPTR$(scale$)
NEXT i
```

Funzioni correlate: PLAY

Istruzioni correlate: ON operazione GOSUB; PLAY (operazione di incanalamento); SOUND

Istruzione PLAY (operazione di incanalamento)

Descrizione

Attiva o disattiva le operazioni di incanalamento.

Sintassi

PLAY ON
oppure
PLAY OFF
oppure
PLAY STOP

Note

- PLAY ON attiva le operazioni di incanalamento.
- PLAY OFF disattiva le operazioni di incanalamento. Tutte le operazioni che vengono effettuate vengono ignorate.
- PLAY STOP sospende temporaneamente le operazioni di incanalamento che verranno elaborate dopo l'attivazione dell'incanalamento.

Esempio

```
ON PLAY GOSUB handler
PLAY ON
```

Funzioni correlate: PLAY

Istruzioni correlate: ON *operazione* GOSUB; PLAY

Funzione PMAP

Descrizione

Restituisce una mappa delle coordinate fisiche a coordinate logiche definite dall'istruzione WINDOW e viceversa.

Sintassi

PMAP(*coordinate*, *mappa*)

Note

- *coordinate* corrisponde a un'espressione numerica delle coordinate su cui deve essere tracciata la mappa.
- *mappa* specifica il tipo di conversione:

Valore	Mappe
0	Coordinate logiche alla *x* fisica.
1	Coordinate logiche alla *y* fisica.
2	Coordinate fisiche alla *x* logica.
3	Coordinate fisiche alla *y* logica.

Esempio

```
SCREEN 1
WINDOW SCREEN (0, 0) -(100, 100)
'Converte logico in fisico
x = PMAP(50, 0)
y = PMAP(50, 1)
```

Istruzioni correlate: VIEW; WINDOW

Funzione POINT

Descrizione

Restituisce le coordinate correnti del cursore grafico o l'attributo di colore di un pixel specificato.

Sintassi

POINT(*x*,*y*)
oppure
POINT(*mappa*)

Note

- La funzione POINT(*x*,*y*) restituisce il colore del pixel nelle coordinate *x*,*y*.
- La funzione POINT(*mappa*) restituisce le coordinate del cursore grafico basate sul valore di *mappa*.

Valore	Riporto
0	Coordinate fisiche *x*.
1	Coordinate fisiche *y*.
2	Coordinate logiche *x*.
3	Coordinate logiche *y*

Esempio

```
x% = POINT(0)
y% = POINT(1)
pcolor% = POINT(x%, y%)
```

Istruzioni correlate: CIRCLE; COLOR; DRAW; LINE; PAINT; PRESET; PSET; SCREEN

Istruzione POKE

Descrizione

Immette un byte in una posizione di memoria specificata.

Sintassi

POKE *offset*, *valore_byte*

Note

- *offset* corrisponde all'offset (da 0 a 65.535) all'interno del segmento attivo in cui si desidera posizionare il byte.
- *byte_valore* corrisponde al valore (da 0 a 255) che verranno inseriti in memoria.

Esempio

```
'Riempire il video CGA con A
DEF SEG= &HB800
FOR i = 0 TO 3999
        IF i MOD 2 = 0 THEN
             POKE i, 65        'lettera A
        ELSE
             POKE i, 7         'attributi video
        END IF
```

```
NEXT i
DEF SEG
```

Funzioni correlate: PEEK

Istruzioni correlate: DEF SEG

Funzione POS

Descrizione

Restituisce la posizione del cursore in colonna.

Sintassi

POS(*argomento_muto*)

Note

- POS non utilizza il parametro *argomento_muto*.

Esempio

```
FOR i = 1 TO 100
       IF (POS(0) > 50 ) THEN
              PRINT i
       ELSE
              PRINT i; 'stessa riga
       END IF
NEXT i
```

Funzioni correlate: CSRLIN; LPOS

Istruzioni correlate: LOCATE

Istruzione PRESET

Descrizione

Disegna un pixel alle coordinate video specificate.

Sintassi

PRESET [**STEP**](*x*,*y*)[,*colore*]

Note

- La parola chiave STEP indica che le coordinate *x* e *y* sono relative alla posizione corrente del cursore grafico.
- PRESET è simile a PSET, con la differenza che, se si omette *colore*, PRESET utilizza il colore di sfondo.
- Se le coordinate sono all'esterno della finestra corrente, non viene disegnato alcun punto.

Esempio

Si veda PSET

Funzioni correlate: POINT

Istruzioni correlate: CIRCLE; COLOR; DRAW; LINE; PAINT; PSET

Istruzione PRINT

Descrizione

Scrive su video o su un file sequenziale

Sintassi

PRINT[#*numero_file*,][*lista_output*][{;|,}]

Note

- *numero_file* corrisponde al numero del file su cui viene scritto l'output. Se si omette *numero_file*, il QBasic scrive i dati a video.
- *lista_output* corrisponde a un elenco di una o più espressioni da emettere che possono essere stringhe o espressioni numeriche.
- Se la riga PRINT termina con un punto e virgola l'istruzione PRINT successiva continua sulla stessa riga nella posizione di carattere successiva. Se la riga viene terminata con una virgola, l'istruzione PRINT successiva continua sulla stessa riga nella zona di stampa successiva, la cui lunghezza massima è di 14 caratteri. Se vengono omesse sia la virgola che il punto e virgola, la successiva istruzione PRINT inizia sulla riga seguente.

Esempio

```
PRINT "Questa è la riga"; 1
PRINT "Questa è la riga";
```

```
PRINT 2
PRINT "Questa è la riga", 3
PRINT "Questa è la riga",
PRINT 4
```

Se si lancia questo programma, viene visualizzato quanto segue:

```
Questa è la riga 1
Questa è la riga 2
Questa è la riga 3
Questa è la riga 4
```

Istruzioni correlate: LPRINT; LPRINT USING; PRINT USING

Istruzione PRINT USING

Descrizione

Scrive un output formattato a video o su un file.

Sintassi

PRINT [#]*numero_file*, **USING** *elenco_formato*; *elenco-output*[{;|,}]

Note

- *numero_file* corrisponde al numero del file su cui viene scritto l'output. Se questo parametro viene omesso, il QBasic scrive i dati a video.
- *elenco_formato* corrisponde a una stringa contenente uno o più dei seguenti indicatori di formato:

Indicatore	Risultato
!	Visualizza solo il primo carattere della stringa.
\\	Stampa 2 + *n* caratteri della stringa, dove *n* corrisponde al numero di spazi tra le barre rovesciate.
&	Visualizza una stringa di qualunque lunghezza.
#	Rappresenta la posizione di una cifra.
.	Rappresenta la posizione del punto decimale.
+	Visualizza un segno + per valori positivi e uno – per i negativi.
–	Aggiunge un segno meno di riempimento a un numero negativo quando questo appare alla fine del campo numerico.

**	Sostituisce spazi di riempimento con asterischi in un campo numerico.
$$	Inserisce il simbolo $ prima di un valore numerico.
**$	Sostituisce spazi di riempimento con asterischi in un campo numerico e inserisce il simbolo $.
,	Inserisce una virgola dopo ogni terza cifra se il simbolo appare a sinistra del punto decimale.
^^^^	Visualizza il valore in formato esponenziale.
_*n*	Stampa il carattere *n* come carattere letterale opposto al carattere formato.
n	Stampa il carattere *n* se non elencato sopra.

Esempio

```
a = 123.4567
PRIN USING "###.##"; a
PRINT USING "+###.####"; a
a$ = "ABCDEFG"
PRINT USING "!"; a$
PRINT USING "\ \"; a$
```

Se si lancia questo programma, viene prodotto il seguente risultato:

```
123.45
+123.4567
A
ABCD
```

Funzioni correlate: LPRINT; LPRINT USING; PRINT

Istruzione PSET

Descrizione

Disegna un pixel alle coordinate video specificate.

Sintassi

PSET[**STEP**](*x*,*y*)[,*colore*]

Note

- La parola chiave STEP indica che le coordinate *x* e *y* sono relative alla posizione corrente del cursore grafico.
- *colore* corrisponde al colore desiderato per il pixel. Se si omette *colore*. PSET utilizza il colore principale attivo.
- Se le coordinate sono all'esterno della finestra corrente, non viene disegnato alcun punto.

Esempio

```
SCREEN 1
COLOR 1,2
CLS
FOR i = 1 TO 10000
     PSET(RND * 320, RND * 200), RND * 4
NEXT i
```

Funzioni correlate: POINT

Istruzioni correlate: CIRCLE; COLOR; DRAW, LINE; PAINT; PRESET; SCREEN

Istruzione PUT (File I/O)

Descrizione

Scrive un buffer di file ad accesso casuale o una variabile record su un file.

Sintassi

PUT [#]*numero_file* [,[*numero_record*][,*variabile*]]

Note

- *numero_file* corrisponde al numero del file assegnato al file nella relativa istruzione OPEN.
- *numero_record* corrisponde al numero del record (da 1 a 2.147.483.647) in un file ad accesso casuale o all'offset di byte in un file binario.
- *variabile* corrisponde a una variabile contenente i campi del record.

Esempio

```
TYPE dipendenti
       dnome AS STRING * 20
       stipendi AS SINGLE
END TYPE

DIM dip AS Dipendenti
OPEN "STIPENDI.DAT" FOR RANDOM AS #1 LEN=LEN(DIP)
dip.nome = "Rossi"
dip.stipendi = 1400000
PUT #1, 1, dip
CLOSE #1
```

Funzioni correlate: CVD; CVDMBF; CVI;CVL;CVS;CVSMBF; MKD$; MKDMBF$; MKI$; MKL$; MKL$; MKSMBF$

Istruzioni correlate: GET; LSET; OPEN; RSET

Istruzione PUT (grafica)

Descrizione

Visualizza un'immagine grafica.

Sintassi

PUT [**STEP**(x,*y*), *array*[(*indice*)][,*azione*]

Note

- La parola chiave STEP indica che le coordinate *x* e *y* sono relative alla posizione corrente del cursore grafico.
- *array* corrisponde al nome dell'array contenente l'immagine che deve essere visualizzata.
- *indice* corrisponde all'indice in cui l'immagine grafica inizia nell'array.
- *azione* indica il modo in cui l'immagine viene visualizzata.

Azione	Risultato
PSET	Disegna un'immagine memorizzata cancellando un'immagine esistente.
PRESET	Disegna un'immagine memorizzata con inversione dei colori cancellando l'immagine precedente.
AND	Unisce l'immagine memorizzata con una precedente.
OR	Sovrappone un'immagine memorizzata a una esistente.
XOR	Disegna un'immagine memorizzata o cancella quella disegnata in precedenza conservando nello stesso tempo lo sfondo. Viene utilizzata per l'animazione.

Esempio

```
SCREEN 1
DIM quadrato(1 TO 200)
x1 = 0
x2 = 10
y1 = 0
y2 =10
LINE (x1, y1)-(x2, y2), 2, BF                'Crea un quadrato
GET (x1, y1)-(x2, y2), quadrato              'Salva l'immagine
FOR i = 1 TO 10000
        PUT (x1, y1), quadrato, XOR          'Sposta il quadrato
        x1 = RND * 300
        y1 = RND * 300                       'sul video
        PUT (x1, y1), quadrato
NEXT i
```

Istruzioni correlate: GET (grafica); SCREEN

Istruzione RANDOMIZE

Descrizione

Inizializza un generatore di numeri casuali.

Sintassi

RANDOMIZE[*seme*]

Note

- *seme* corrisponde a un'espressione che inizializza il generatore di numeri casuali. Se viene omesso, RANDOMIZE richiede all'utente di immetterlo.

Esempio

```
'stampa 50 numeri a caso
'utilizzando 10 origini differenti
FOR i = 1 TO 10
      PRINT "Origine"; i
      RANDOMIZE i
      FOR j = 1 TO 5
          PRINT RND
      NEXT j
NEXT i
```

Funzioni correlate: RND; TIMER

Istruzione READ

Descrizione

Legge i valori da un elenco di dati e li assegna a variabili.

Sintassi

READ *elenco_variabili*

Note

- *elenco_variabili* corrisponde a un elenco di variabili tra loro separate da virgole. Le istruzioni READ e DATA lavorano insieme per assegnare valori a queste variabili.

Esempio

```
READ n$, età, stipendi
READ indirizzo$
DATA "Colombo", 21, 500
DATA "Milano"
PRINT n$, età, stipendi
PRINT indirizzo$
```

Istruzioni correlate: DATA; RESTORE

Istruzione REDIM

Descrizione

Modifica le dimensioni di un array dinamico.

Sintassi

REDIM [**SHARED**]*array*(*subscripts*)[**AS** *tipo*][,*array*(*subscript*)[**AS** *tipo*]...

Note

- È possibile modificare le dimensioni dell'array dinamico ma non il numero delle dimensioni o del tipo di dati.
- La parola chiave SHARED indica che l'array può essere utilizzato con tutte le procedure nel modulo.
- *array* corrisponde al nome dell'array le cui dimensioni devono essere modificate.
- *subscripts* specifica i subscripts dell'array nel formato: [*boundinferiore* **TO**]*boundsuperiore*
- Vengono gestiti gli array multidimensionali.
- *tipo* corrisponde al tipo di array: INTEGER, LONG, SINGLE, DOUBLE, STRING oppure un tipo definito dall'utente.
- REDIM cancella i valori precedenti dell'array.

Esempio

```
'$DYNAMIC
DIM box (1 TO 100)
'istruzione
REDIM box (1 TO 200)
```

Istruzioni correlate: DIM; ERASE

Istruzione REM

Descrizione

Permette di inserire un'indicazione su una riga di programma.

Sintassi

REM *indicazione*

Note

- *indicazione* corrisponde a un qualunque testo.
- Il QBasic ignora le indicazioni a meno che esse non contengano metacomandi di compilatore.
- Il QBasic gestisce l'utilizzo di una sola virgoletta (') al posto dell'istruzione REM.

Esempio

```
REM Questo è un'indicazione
'Questa è un'indicazione
```

Istruzione RESET

Descrizione

Scrive su disco qualsiasi dato ancora nei buffer di file e chiude tutti i file su disco.

Sintassi

RESET

Note

- RESET chiude tutti i file in una volta. È possibile utilizzare l'istruzione CLOSE per chiudere singolarmente ciascun file.

Esempio

```
RESET
```

Istruzioni correlate: CLOSE; OPEN

Istruzione RESTORE

Descrizione

Permette a READ di utilizzare nuovamente l'istruzione DATA precedentemente letta.

Sintassi

RESTORE [*posizione*]

Note

- *posizione* corrisponde al numero di riga o all'etichetta dell'istruzione DATA che deve essere letta. SE *posizione* viene omessa, l'istruzione READ successiva utilizza la prima istruzione DATA del programma.

Esempio

```
FOR i = 1 TO 5
        READ a%, b%, c%
        PRINT a%, b%, c%
        RESTORE
NEXT i
DATA 1, 2, 3
```

Istruzioni correlate: DATA; READ

Istruzione RESUME

Descrizione

Prosegue l'esecuzione di un programma dopo aver avviato un routine di gestione degli errori.

Sintassi

RESUME [*posizione*]

oppure

RESUME NEXT

Note

- *posizione* corrisponde al numero di riga o all'etichetta da cui l'esecuzione dovrebbe proseguire. Se si specifica 0 o si omette *posizione*, l'esecuzione prosegue partendo dall'istruzione e genera un messaggio di errore.
- La parola chiave NEXT prosegue l'esecuzione dall'istruzione immediatamente successiva, generando così un messaggio di errore.

Esempio

```
ON ERROR GOTO Handler
OPEN "A:TEST.DAT" FOR INPUT AS #1
'istruzione
END

Handler:
   PRINT "Inserire il disco contenente TEST.DAT
   PRINT "nell'unità A:. Premere Invio
   INPUT muto$
RESUME
```

Istruzioni correlate: ERROR; ON ERROR GOTO

Istruzione RETURN

Descrizione

Restituisce il controllo da una subroutine alla procedura di chiamata.

Sintassi

RETURN [*posizione*]

Note

- *posizione* corrisponde al numero di riga o all'etichetta da cui l'esecuzione dovrebbe continuare. Se si omette *posizione*, l'esecuzione prosegue dalla riga seguente l'istruzione GOSUB, o per la gestione delle operazioni, dalla riga in cui l'operazione viene eseguita.

Esempio

```
GOSUB uno
GOSUB due
END
```

```
uno:
       PRINT "In uno"
       RETURN

due:
       PRINT "In due"
       RETURN
```

Istruzioni correlate: GOSUB; ON operazione GOSUB

Funzione RIGHT$

Descrizione

Restituisce il numero specificato di caratteri partendo dal carattere più a sinistra della stringa.

Sintassi

RIGHT$(*stringa*, *num_car*)

Note

- *stringa* corrisponde a una qualsiasi stringa.
- *num_car* corrisponde al numero di caratteri da restituire. Se *num_car* è più lungo della stringa, RIGHT$ restituisce la stringa intera.

Esempio

```
A$ = "ABCDEFGHIJ"
FOR I = 1 TO 10
        PRINT RIGHT$(A$, I)
NEXT I
```

Istruzioni correlate: INSTR; LEFT$; LEN; MID$

Istruzione RMDIR

Descrizione

Cancella la directory specificata.

Sintassi

RMDIR *nome_directory*

Note

- *nome_directory* corrisponde alla stringa contenente il nome della directory da cancellare.
- RMDIR funziona come il comando DOS Remove Directory. Non può cancellare la directory corrente o un qualsiasi directory che contenga file.

Esempio

```
RMDIR "A:\TEST"
```

Istruzioni correlate: CHDIR; MKDIR

Funzione RND

Descrizione

Restituisce un numero casuale a precisione singola tra 0 e 1.

Sintassi

RND[(*espressione_numerica*)]

Note

- *espressione_numerica* specifica come RND genera il numero casuale successivo:

Valore	**Risultato**
Minore di 0	Lo stesso numero per qualunque *espressione_numerica* data.
Uguale a 0	L'ultimo numero creato.
Maggiore di 0	Il numero casuale successivo.

- Se si omette *espressione_numerica*, RND genera il numero successivo in sequenza.

Esempio

```
'stampa 10 numeri casuali
FOR i = 1 TO 10
```

```
        PRINT INT(RND * 10)
NEXT i
```

Istruzioni correlate: RANDOMIZE

Istruzione RSET

Descrizione

Sposta i dati all'interno di un buffer di file ad accesso casuale o giustifica a destra il valore di una variabile stringa.

Sintassi

RSET *variabile_stringa* = *stringa*

Note

- *variabile_stringa* corrisponde sia a una variabile di file ad accesso casuale sia a una variabile stringa.
- Per le variabili di file ad accesso casuale, RSET assegna un tipo di variabile di record a un altro.
- Per le variabili stringa, RSET giustifica a destra la stringa.

Esempio

```
DIM n AS STRING * 10
PRINT "ABCDE"
RSET n = "ABCDE"
PRINT n
```

Se si lancia questo programma, viene visualizzato quanto segue:

```
ABCDE
     ABCDE
```

Istruzioni correlate: FIELD; LSET

Funzione RTRIM$

Descrizione

Cancella gli spazi di riempimento da una stringa.

Sintassi

RTRIM$(*stringa*)

Note

- *stringa* corrisponde a una qualsiasi stringa.

Esempio

```
a$ = "AAAA    "
b$ = "BBBB"
PRINT RTRIM$(a$); b$
```

Se si lancia questo programma, viene visualizzato quanto segue:

```
AAAABBBB
```

Funzioni correlate: LTRIM$

Istruzione RUN

Descrizione

Avvia il programma corrente in memoria o un programma esistente da disco.

Sintassi

RUN[*numero_riga*]

oppure

RUN[*numero_file*]

Note

- *numero_riga* corrisponde al numero di riga nel programma attivo da cui deve iniziare l'esecuzione; se viene omesso, RUN inizia dal primo numero di riga.

- *nome_file* corrisponde a un'espressione contenente il nome del file da eseguire. Il QBasic assume per default l'estensione .BAS.
- RUN chiude tutti i file e cancella tutte le variabili. Per condividere le variabili, si utilizzi l'istruzione CHAIN.

Esempio

```
RUN "NOMEFILE"
```

Istruzioni correlate: CHAIN

Funzione SCREEN

Descrizione

Restituisce il carattere o l'attributo colore per il carattere specificato alla colonna e alla riga.

Sintassi

SCREEN(*riga*, *colonna*[,*acc_colore*])

Note

- *riga* e *colonna* corrispondono alle coordinate del carattere.
- *acc_colore* corrisponde a un'espressione numerica. Se *acc_colore* è uguale a 1, SCREEN restituisce il colore del carattere; se *acc_colore* è uguale a 0 o viene omesso, SCREEN restituisce il valore ASCII del carattere alla posizione specificata.

Esempio

```
DIM scr(1 TO 25, 1 TO 80) AS INTEGER
'registra il contenuto attivo dello schermo
FOR row = 1 TO 25
        FOR column = 1 TO 80
            scr(row, column) = SCREEN(row, column)
        NEXT column
NEXT row
```

Istruzione SCREEN

Descrizione

Definisce la caratteristiche dello schermo.

Sintassi

SCREEN [*modalità*][,[*coloreoff*][,[*pagina_attiva*][,*pagina_visualizzata*]]]

Note

- *modalità* corrisponde a un'espressione intera che specifica la modalità dell'operazione:

Valore	Modo	Adattatore
0	Testo	CGA, EGA, VGA, MCGA
1	Grafica 320 x 200	CGA, EGA, VGA, MCGA
2	Grafica 640 x 200	EGA, VGA
3	Grafica 720 x 348	Hercules*
4	Grafica 640 x 400	Olivetti, AT&T 6300
7	Grafica 320 x 200	EGA, VGA
8	Grafica 640 x 200	EGA, VGA
9	Grafica 640 x 350	EGA, VGA
10	Grafica 640 x 350	EGA, VGA
11	Grafica 640 x 480	VGA, MCGA
12	Grafica 640 x 480	VGA
13	Grafica 320 x 200	VGA, MCGA

* *Il driver Hercules MSHER.COM deve essere caricato.*

- *coloreoff* corrisponde a un'espressione numerica che, se è vera, disattiva il colore su un monitor composito (questo parametro viene ignorato nelle modalità schermo dalla 2 in avanti).
- *pagina_attiva* corrisponde alla pagina di schermo nella quale vengono scritti un testo o un grafico.
- *pagina_visualizzata* corrisponde alla pagina di schermo attualmente visualizzata.

Esempio

```
SCREEN 1     '320 x 200 grafica
LINE (10, 10)-(20, 20), , B
```

Funzioni correlate: POINT

Istruzioni correlate: CIRCLE; COLOR; DRAW; GET (grafica); LINE; PAINT; PALETTE; PALETTE USING; PRESET; PSET: PUT (grafica); VIEW; WINDOW

Funzione SEEK

Descrizione

Restituisce la posizione del puntatore del file attivo.

Sintassi

SEEK (*numero_file*)

Note

- *numero_file* corrisponde al numero assegnato al file nella relativa istruzione OPEN.
- Per i file ad accesso casuale, SEEK restituisce un numero di record compreso tra 1 e 2.147.483.647. Per i file binari e sequenziali, SEEK restituisce l'offset di byte corrente.

Esempio

```
position = SEEK(1)
```

Funzioni correlate: LOC

Istruzioni correlate: OPEN; SEEK

Istruzione SEEK

Descrizione

Imposta la posizione del puntatore del file per la successiva operazione di lettura o scrittura.

Sintassi

SEEK[#]*numero_file*, *posizione*

Note

- *numero_file* corrisponde al numero assegnato al file nell'istruzione OPEN.
- *posizione* corrisponde al numero di record desiderato in un file ad accesso casuale o all'offset di byte in un file binario o sequenziale. Deve essere compreso tra 1 e 2.147.483.647.

Esempio

```
OPEN "STIPENDI.DAT" FOR RANDOM AS #1 LEN = 80
SEEK #1, 5       'Sposta il puntatore al record 5
```

Funzioni correlate: LOC; SEEK

Istruzioni correlate: GET (FileI/O); OPEN; PUT (File I/O)

Istruzione SELECT CASE

Descrizione

Analizza un'espressione ed esegue il corrispondente blocco di istruzioni.

Sintassi

SELECT CASE *testo*
CASE *testo_corrispondente*
  [*istruzioni*]
[**CASE** *testo_corrispondente*
  [*istruzioni*]]
...
[**CASE ELSE**
  [*istruzioni_default*]]
END SELECT

Note

- L'istruzione SELECT CASE analizza un'espressione e ricerca un caso corrispondente nell'elenco. Se questo viene trovato, il QBasic esegue l'istruzione per quel caso.

- *testo* corrisponde a una stringa o a un'espressione numerica che viene analizzata e confrontata con i casi possibili.

- *testo_corrispondente* corrisponde a un'espressione che deve equivalere a *testo*. Può avere il seguente formato:
  testo[,*testo*]...
  Se uno qualunque dei testi elencati in *testo* corrisponde, viene eseguita l'operazione indicata in *istruzioni*.

- *testo_corrispondente* può inoltre avere il seguente formato:
  testo **TO** *testo*
  Questo fornisce un intervallo di possibili valori che verranno confrontati.

- Infine, *testo_corrispondente* può avere il formato:
  IS *operatore espressione*
  dove l'operatore è <, >, <=, >=, =, oppure <>.

- *istruzioni* corrisponde all'elenco di istruzioni che vengono eseguite se *testo* trova *testo_corrispondente*.

- *istruzioni_default* corrisponde all'elenco di istruzioni che vengono eseguite se non viene trovato un testo corrispondente. Queste istruzioni vengono associate alla condizione CASE ELSE.

- Dopo che sono state eseguite le istruzioni all'interno del testo corrispondente, il programma prosegue eseguendo la prima istruzione successiva all'istruzione END SELECT.

Esempio

```
FOR i = 1 TO 5
     SELECT CASE i
     CASE 1
          PRINT "uno"
     CASE 2, 3
          PRINT "due o tre"
     CASE IS = 4
          PRINT "quattro"
     CASE ELSE
          PRINT "cinque"
     END SELECT
NEXT i
```

Se si lancia questo programma, viene visualizzato quanto segue:

```
uno
due o tre
due o tre
quattro
cinque
```

Istruzioni correlate: IF

Funzione SGN

Descrizione

Restituisce un valore indicante il segno di un'espressione.

Sintassi

SGN(*espressione_numerica*)

Note

- *espressione_numerica* corrisponde a un'espressione numerica qualsiasi. Se il valore dell'espressione è positivo, SGN restituisce –1 e se il valore è 0 restituisce 0.

Esempio

```
PRINT SGN(-3)
PRINT SGN(0)
PRINT SGN(127)
```

Se si lancia questo programma, viene visualizzato quanto segue:

```
-1
 0
 1
```

Istruzione SHARED

Descrizione

Consente a un sottoprogramma o a una funzione di accedere alle variabili del modulo.

Sintassi

SHARED *variabile*[**AS** *tipo*][,*variabile*[**AS** *tipo*]]...

Note

- Per default, un sottoprogramma o una funzione ha accesso a una variabile solo se la variabile viene considerata come un parametro.
- *variabile* corrisponde al nome della variabile del modulo da condividere.
- *tipo* corrisponde al nome del tipo di variabile: INTEGER, LONG, SINGLE, DOUBLE, STRING o un tipo di variabile definito dall'utente.

Esempio

```
DIM a AS INTEGER
a = 5
CALL Test
END
SUB Test
       SHARED a AS INTEGER
       PRINT "Il valore della variabile A è"; a
END SUB
```

Istruzione correlata: DIM

Istruzione SHELL

Descrizione

Sospende temporaneamente il programma per eseguire un comando DOS o un file batch.

Sintassi

SHELL [*comando_DOS*]

Note

- *comando_DOS* corrisponde a una stringa che specifica il comando da eseguire. Dopo che il comando DOS è stato eseguito, il programma prosegue. Se si omette *comando_DOS*, SHELL visualizza il prompt del DOS. Una volta terminato di lavorare con il DOS, si utilizzi il comando DOS Exit per riavviare il programma.

Esempio

```
SHELL "dir"                    'Visualizza un elenco di file

SHELL                           'prompt MS-DOS
```

Funzione SIN

Descrizione

Restituisce il seno di un angolo specificato.

Sintassi

SIN(*angolo*)

Note

- *angolo* corrisponde a un'espressione numerica che specifica un angolo in radianti.
- L'angolo può essere espresso in radianti o gradi. La trigonometria di QBasic riconosce solo i radianti. Per convertire i gradi in radianti si utilizzi la seguente equazione:
  radianti = 3.141593 * (*gradi*/180)

Esempio

```
pi = 3.141593
PRINT "Seno di pi", SIN(pi)
PRINT "Seno di pi/2", SIN(pi / 2)
```

Se si lancia questo programma, viene visualizzato quanto segue:

```
Seno di pi        -3.258414E-07
Seno di pi/2       1
```

Funzioni correlate: ATN; COS; TAN

Funzione SLEEP

Descrizione

Sospende l'esecuzione del programma per l'intervallo di tempo specificato.

Sintassi

SLEEP [*secondi*]

Note

- *secondi* corrisponde al numero di secondi in cui il programma verrà sospeso.
- Il programma rimane sospeso e viene ripreso solo se l'utente preme un tasto, quando termina il numero di secondi indicato o se si verifica un'operazione di incanalamento.
- Se si omette *secondi* o questo valore è minore di 1, il programma resta sospeso fino a quando l'utente non preme un tasto o non si verifica un'operazione di incanalamento.

Esempio

```
SLEEP 30
```

Istruzione SOUND

Descrizione

Indica al computer di emettere un segnale sonoro.

SOUND *frequenza*, *durata*

Note

- *frequenza* corrisponde a un'espressione intera (da 37 a 32.767) che specifica la frequenza del suono in hertz.
- *durata* corrisponde a un'espressione intera non segnata (da 0 a 65,535) che specifica la lunghezza del suono in impulsi clock. Un impulso clock viene emesso 18.2 volte al secondo.

Esempio

```
FOR i = 37 TO 3000
     PRINT i
SOUND i, 1
NEXT i
```

Funzioni correlate: PLAY

Istruzioni correlate: BEEP; PLAY

Funzione SPACE$

Descrizione

Restituisce una stringa contenente il numero specificato di spazi.

Sintassi

SPACE$(*numero_spazi*)

Note

- *num_spazi* corrisponde a un'espressione intera (da 0 a 32.767).

Esempio

```
FOR i = 0 TO 5
    PRINT SPACE$(i); i
NEXT i
```

Se si lancia questo programma, viene visualizzato quanto segue:

```
0
 1
  2
   3
    4
     5
```

Funzioni correlate: SPC; TAB

Funzione SPC

Descrizione

Salta il numero specificato di spazi in un'istruzione PRINT o LPRINT.

Sintassi

SPC(*numero_spazi*)

Note

- *numero_spazi* corrisponde a un valore intero (da 0 a 32.767).

Esempio

```
FOR i = 0 TO 5
       PRINT SPC(i); i
NEXT i
```

Funzioni correlate: SPACE$; TAB

Funzione SQR

Descrizione

Restituisce la radice quadrata di un'espressione.

Sintassi

SQR(*espressione_numerica*)

Note

- *espressione_numerica* corrisponde a una qualsiasi espressione numerica non negativa.

Esempio

```
FOR i = 0 TO 100
       PRINT i, SQR(i)
NEXT i
```

Istruzione STATIC

Descrizione

Rende locale la variabile specificata per un sottoprogramma o una funzione e indica al QBasic di mantenere il valore della variabile tra le chiamate.

Sintassi

STATIC *variabile*[**AS** *tipo*][,*variabile*[**AS** *tipo*]]...

Note

- *variabile* corrisponde al nome della variabile che verrà trasformata in statica.

- *tipo* corrisponde al tipo di variabile: INTEGER, LONG, SINGLE, DOUBLE STRING o una variabile definita dall'utente.

Esempio

```
CALL test
CALL Test
END

SUB Test
    STATIC a AS INTEGER
    PRINT a
    a = a + 1
END SUB
```

Se si lancia questo programma, viene visualizzato quanto segue:

```
0
1
```

Istruzioni correlate: COMMON; SHARED

Funzione STICK

Descrizione

Restituisce le coordinate *x* o *y* di un joystick.

Sintassi

STICK(*espressione_numerica*)

Note

- *espressione_numerica* corrisponde a un valore intero senza segno compreso tra 0 e 3 che specifica il valore desiderato.

Valore	Riporto
0	coordinate *x* del joystick A.
1	coordinate *y* del joystick A.
2	coordinate *x* del joystick B.
3	coordinate *y* del joystick B

- Le coordinate *x* e *y* possono essere comprese tra 1 e 200.

Esempio

```
x% = STICK(0)
y% = STICK(1)
```

Istruzione STOP

Descrizione

Termina il programma in qualunque punto.

Sintassi

STOP

Note

- Un programma dovrebbe avere un solo punto di inizio e uno di fine: si sconsiglia pertanto di utilizzare STOP in un diverso punto del programma.

Esempio

```
Handler:
    PRINT "Non è stato possibile aprire il file
    PRINT "al terzo tentativo"
    STOP
```

Funzione STR$

Descrizione

Restituisce un'espressione numerica in forma di stringa.

Sintassi

STR$(*espressione_numerica*)

Note

- *espressione_numerica* corrisponde a una qualsiasi espressione.

Esempio

```
x$ = STR$(3,2718)
PRINT x$
```

Funzioni correlate: VAL

Funzione STRIG

Descrizione

Restituisce lo stato del pulsante di azione del joystick.

Sintassi

STRIG(*espressione_numerica*)

Note

- *espressione_numerica* corrisponde a un intero non segnato (da 0 a 7) che specifica il tipo di informazione desiderata:

Valore	Condizione
0	Il pulsante inferiore del joystick A è stato premuto dall'ultima volta in cui è stato avviato STRIG(0).
1	Il pulsante inferiore del joystick A è attualmente premuto.
2	Il pulsante inferiore del joystick B è stato premuto dopo l'ultimo STRIG(2)
3	Il pulsante inferiore del joystick B è attualmente premuto.
4	Il pulsante superiore del joystick A è stato premuto dopo l'ultimo STRIG(4).
5	Il pulsante superiore del joystick A è attualmente premuto.
6	Il pulsante superiore del joystick B è stato premuto dall'ultimo STRIG(6).
7	Il pulsante superiore del joystick B è attualmente premuto.

- Se la condizione specificata viene soddisfatta, STRIG restituisce –1; altrimenti restituisce 0.

Esempio

```
'L'utente deve premere
'il pulsante inferiore del joystick A
DO
LOOP UNTIL STRIG(0)
```

Istruzioni correlate: ON operazione GOSUB; STRIG

Istruzione STRIG

Descrizione

Attiva, disattiva o sospende la gestione delle operazioni del joystick.

Sintassi

STRIG(*pulsante*) **ON**
oppure
STRIG(*pulsante*) **OFF**
oppure
STRIG(*pulsante*) **STOP**

Note

- *pulsante* corrisponde a un'espressione numerica che specifica quale pulsante del joystick viene incanalato:

Valore	Incanalamento
0	Pulsante inferiore del joystick A
2	Pulsante inferiore del joystick B
4	Pulsante superiore del joystick B
6	Pulsante superiore del joystick B

- STRIG(*pulsante*) ON attiva l'incanalamento per il pulsante specificato.
- STRIG(*pulsante*) OFF disattiva l'incanalamento del joystick per il pulsante specificato. Tutte le operazioni vengono ignorate.
- STRIG(*pulsante*) STOP disattiva temporaneamente l'incanalamento del joystick per il pulsante specificato. Le operazioni verranno elaborate dopo che ne sarà stata attivata la gestione con STRIG ON.

Esempio

```
ON STRIG(0) GOSUB Handler
STRIG(0) ON
```

Funzioni correlate: STRIG

Istruzioni correlate: ON *operazione* GOSUB

Funzione STRING$

Descrizione

Restituisce una stringa contenente il numero specificato in cui appare un carattere.

Sintassi

STRING$(*numero*, *car_ascii*)
oppure
STRING$(*numero*, *stringa*)

Note

- *numero* corrisponde al numero di volte in cui si desidera che appaia il carattere.
- *car_ascii* corrisponde al codice ASCII del carattere.
- *stringa* corrisponde a una qualsiasi stringa. Se si fornisce una stringa, STRNG$ utilizza il primo carattere di una stringa.

Esempio

```
a$ = STRING$(10, 65)
PRINT a$
```

Istruzione SUB

Descrizione

Dichiara un sottoprogramma Basic.

Sintassi

SUB *nome_sottoprogramma* [(*elenco_parametri*)] [**STATIC**]
...
END SUB

Note

- *nome_sottoprogramma* corrisponde al nome del sottoprogramma (fino a 40 caratteri).
- *elenco_parametri* corrisponde a un elenco di parametri nel seguente formato: *variabile*[()][**AS** *tipo*][,*variabile*[()][**AS** *tipo*]]...
- La parola chiave STATIC indica a QBasic di mantenere il valore delle variabili locali del sottoprogramma tra le chiamate.

Esempio

```
CALL Test (1, 5.5, "TEST")
END

SUB Test (a AS INTEGER, b AS SINGLE, c AS STRING)
      PRINT a, b, c
END SUB
```

Istruzioni correlate: CALL; DECLARE; END; EXIT; GOSUB

Istruzione SWAP

Descrizione

Scambia il valore di due variabili.

Sintassi

SWAP *variabile1*, *variabile2*

Note

- *variabile1* e *variabile2* devono essere dello stesso tipo.

Esempio

```
a = 1
b = 2
```

```
SWAP a, b
PRINT a, b
```

Se si lancia questo programma, viene visualizzato quanto segue:

```
2   1
```

Istruzione SYSTEM

Descrizione

Termina un programma e restituisce il controllo al sistema operativo.

Sintassi

SYSTEM

Note

- SYSTEM chiude tutti i file aperti e termina l'esecuzione del programma.

Esempio

```
SYSTEM
```

Funzione TAB

Descrizione

Sposta la posizione di stampa alla colonna specificata.

Sintassi

TAB(*colonna*)

Note

- *colonna* corrisponde alla colonna di tabulazione desiderata. Se la posizione corrente è al di là della colonna specificata, TAB sposta la colonna alla riga successiva.

Esempio

```
FOR i = 1 TO 10
   PRINT TAB(i); i
NEXT i
```

Funzioni correlate: SPC

Funzione TAN

Descrizione

Restituisce la tangente di un angolo specificato.

Sintassi

TAN(*angolo*)

Note

- *angolo* corrisponde a un'espressione numerica che specifica l'angolo in radianti.
- È possibile esprimere un angolo in radianti o in gradi. Le routine di trigonometria del QBasic gestiscono solo radianti.
  Per convertire i gradi in radianti, si utilizzi la seguente equazione:

  radianti = 3, 141593 * (*gradi* / 180)

Esempio

```
pi = 3,141593
PRINT TAN(pi / 4)
```

Funzioni correlate: ATN; COS; SIN

Funzione TIME$

Descrizione

Restituisce una stringa di 8 caratteri contenente l'ora di sistema corrente nel formato *hh:mm:gg*.

Sintassi

TIME$

Esempio

```
PRINT TIME$
```

Se si lancia questo programma a mezzogiorno viene visualizzato quanto segue:

```
12:00:00
```

Funzioni correlate: DATE$

Istruzioni correlate: DATE$; TIME$

Istruzione TIME$

Descrizione

Imposta l'ora di sistema.

Sintassi

TIME$=*stringa*

Note

- *stringa* corrisponde alla stringa contenete l'ora desiderata nel formato *hh:mm:ss* dove *hh* è l'ora (da 0 a 23), *mm* sono i minuti (da 0 a 59), e *ss* sono i secondi (da 0 a 59).
- TIME$ consente di specificare solo l'ora, o solo l'ora e i minuti oppure l'ora, i minuti e i secondi.

Esempio

```
TIME$ = "12"
TIME$ = "12:30"
```

Funzioni correlate: DATE$; TIME$

Istruzioni correlate: DATE$

Funzione TIMER

Descrizione

Restituisce il numero di secondi da mezzanotte.

Sintassi

TIMER

Note

- È possibile utilizzare TIMER con l'istruzione RANDOMIZE per generare un numero casuale.

Esempio

```
RANDOMIZE TIMER
```

Istruzione TIMER

Descrizione

Attiva o disattiva le operazioni di incanalamento del timer.

Sintassi

TIMER ON
oppure
TIMER OFF
oppure
TIMER STOP

Note

- TIMER ON attiva le operazioni di incanalamento.
- TIMER OFF disattiva le operazioni di incanalamento. Le operazioni di timer vengono ignorate.
- TIMER STOP sospende temporaneamente l'incanalamento delle operazioni timer. Le operazioni che si presentano vengono eseguite quando l'incanalamento viene attivato.

Esempio

```
ON TIMER(10)  GOSUB Handler
TIMER ON
DO
LOOP UNTIL INKEY$ <> ""
END

Handler:
   PRINT TIME$
   RETURN
```

Istruzioni correlate: ON *operazione* GOSUB

Istruzione TROFF

Descrizione

Disattiva le istruzioni di traccia.

Sintassi

TROFF

Note

- TROFF e TRON sono comandi di debugging utilizzati da vecchi sistemi Basic. Si noterà spesso che le operazioni di debugging del QBasic sono più pratiche.

Esempio

```
TROFF
```

Funzioni correlate: TRON

Istruzione TRON

Descrizione

Attiva la traccia delle istruzioni.

Sintassi

TRON

Note

- Si veda TROFF

Esempio

```
TRON
```

Istruzione correlata: TROFF

Istruzione TYPE

Descrizione

Crea un tipo di dati definito dall'utente.

Sintassi

TYPE *tipo_utente*
  nome_elemento **AS** *tipo*
  ...
END TYPE

Note

- *tipo_utente* corrisponde al nome del tipo definito dall'utente.
- *nome_elemento* corrisponde al nome dell'elemento in un record.
- *tipo* corrisponde al tipo di elemento: INTEGER, LONG, SINGLE, DOUBLE, la lunghezza fissata per la stringa (per esempio STRNG * 4) o un altro tipo definito dall'utente.
- Per creare una variabile di un tipo di dati definito dall'utente è necessario utilizzare DIM, REDIM, COMMON, STATIC o SHARED.

Esempio

```
TYPE Dipendente
      dnome AS STRING * 20
      stipendi AS SINGLE
END TYPE

DIM dip AS Dipendente
```

```
dip.nome = "Mapelli"
dip.stipendi = 1100000
PRINT dip.nome, dip.stipendi
```

Se si lancia questo programma, verrà visualizzato quanto segue:

```
Mapelli    1100000
```

Istruzioni correlate: COMMON; DIM, REDIM; SHARED; STATIC

Funzione UBOUND

Descrizione

Restituisce il più alto indice inferiore di un array in base alle dimensioni specificate dell'array.

Sintassi

UBOUND(*nome_array*[,*dimensioni*])

Note

- *nome_array* corrisponde al nome dell'array di interesse.
- *dimensioni* corrisponde a un valore intero che specifica le dimensioni desiderate in un array multidimensionale. Il valore impostato per default è 1.

Esempio

```
DIM a(1 TO 5, 1 TO 10, 1 TO 25)
PRINT UBOUND(a), UBOUND(a, 2), UBOUND(a, 3)
```

Se si lancia questo programma, viene visualizzato quanto segue:

```
5      10      25
```

Funzioni correlate: LBOUND

Istruzioni correlate: DIM; OPTION BASE; REDIM

Funzione UCASE$

Descrizione

Restituisce una stringa con tutte le lettere maiuscole.

Sintassi

UCASE$(*stringa*)

Note

- *stringa* corrisponde a una qualsiasi stringa.

Esempio

```
a$ = "aBcd#F"
PRINT UCASE$(a$)
```

Se si lancia questo programma, viene visualizzato quanto segue:

```
ABCD#F
```

Funzioni correlate: LCASE$

Istruzione UNLOCK

Descrizione

Apre le sezioni di un file condiviso affinché possano accedervi programmi da altre reti eliminando le limitazione imposte con l'ultima istruzione LOCK.

Sintassi

UNLOCK[#]*numero_file*[,{*record* |[*inizio*] **TO** *fine*}]

Note

- *numero_file* corrisponde al numero assegnato al file nella relativa istruzione OPEN.

- Per i file ad accesso casuale, *record* corrisponde al numero di un record da bloccare in base al primo record nel file; per i file binari corrisponde al numero di un byte da bloccare in base al primo byte nel file.

- *inizio* e *fine* corrispondono ai numeri del primo e dell'ultimo record o byte all'interno di un intervallo di record o byte da bloccare o da sbloccare.
- UNLOCK è necessario solo in ambienti a rete.

Esempio

```
OPEN "SHARED.DAT" FOR RANDOM AS #1
LOCK #1, 1 TO 10
'Eseguire le operazioni di aggiornamento
UNLOCK #1, 1 TO 10
CLOSE #1
```

Istruzioni correlate: LOCK

Funzione VAL

Descrizione

Riconverte un numero in forma di stringa.

Sintassi

VAL(*stringa*)

Note

- *stringa* corrisponde al numero in forma di stringa.
- VAL si ferma al primo carattere che non può riconoscere come parte del numero. Vengono considerati caratteri validi da 0 a 9 , il punto (.), il segno meno (–), e il segno più (+).

Esempio

```
PRINT VAL("33.44")
PRINT VAL("88k")
```

Se si lancia questo programma, verrà visualizzato quanto segue:

```
33.44
88
```

Funzioni correlate: STR$

Funzione VARPTR

Descrizione

Restituisce l'indirizzo dell'offset in una variabile.

Sintassi

VARPTR(*variabile*)

Note

- *variabile* corrisponde al nome di una qualsiasi variabile del programma.
- Il QBasic non garantisce che la variabile si trovi nella stessa locazione di memoria durante l'esecuzione del programma. Si utilizzi VARPTR subito prima di un qualsiasi codice che utilizza il valore di offset.

Esempio

Si veda CALL ABSOLUTE

Funzioni correlate: VARSEG

Istruzioni correlate: DEF SEG

Funzione VARPTR$

Descrizione

Restituisce l'indirizzo di una variabile in forma di stringa che può essere utilizzata con le istruzioni PLAY e DRAW.

Sintassi

VARPTR$(*variabile_stringa*)

Note

- *variabile_stringa* corrisponde a una variabile stringa contenente i comandi DRAW o PLAY.
- Il QBasic non garantisce che la variabile si trovi nella stessa locazione di memoria durante l'esecuzione del programma. Si utilizzi VARPTR$ subito prima di un qualsiasi codice che utilizzi l'indirizzo.

Esempio

```
scale$ = "CDEFGAB"
PLAY "X" + VARPTR$(scale$)
```

Istruzioni correlate: DRAW; PLAY

Funzione VARSEG

Descrizione

Restituisce l'indirizzo del segmento di una variabile.

Sintassi

VARSEG(*variabile*)

Note

- *variabile* corrisponde al nome di una qualsiasi variabile di programma.

Esempio

Si veda CALL ABSOLUTE.

Funzioni correlate: VARPTR

Istruzioni correlate: DEF SEG

Istruzione VIEW

Descrizione

Definisce le coordinate video all'interno del quale verrà visualizzato il grafico.

Sintassi

VIEW[[**SCREEN**](*x1*, *y1*)-(*x2*,*y2*)[,[*colore_riempimento*][,*bordo_presente*]]]

Note

- VIEW permette di limitare l'emissione della grafica per specificare le coordinate a video. Le coordinate non comprese in questo intervallo non verranno disegnate.

- La parola chiave SCREEN indica che le coordinate dei grafici sono relative al video, non alla finestra.
- *x1* e *y1* corrispondono alle coordinate di un angolo della finestra. *x2* e *y2* sono le coordinate dell'angolo opposto.
- *colore_riempimento* specifica il colore con cui deve essere riempita la finestra.
- *bordo_presente* corrisponde a qualsiasi espressione numerica che, quando presente, indica a VIEW di disegnare un bordo attorno alla finestra.
- Se si omettono tutti gli argomenti, VIEW imposta la finestra a schermo intero.
- Le istruzioni SCREEN e RUN impostano nuovamente la finestra a schermo intero.

Esempio

```
SCREEN 1
VIEW (0,0)-(20, 20), 1, 2
LINE (10, 10)-(100, 100)
```

Istruzioni correlate: CLS; SCREEN; WINDOW

Istruzione VIEW PRINT

Descrizione

Definisce i limiti della finestra testo dello schermo.

Sintassi

VIEW PRINT [*riga_superiore* **TO** *riga_inferiore*]

Note

- *riga_superiore* e *riga_inferiore* corrispondono a valori interi che specificano le righe superiori e inferiori della regione di scorrimento della modalità testo.
- Se tutti gli argomenti vengono omessi, VIEW PRINT imposta la regione di scorrimento all'intero schermo.

Esempio

```
CLS
VIEW PRINT 5 TO 10
```

```
FOR i = 1 TO 100
     PRINT i, i, i, i
NEXT i
```

Istruzioni correlate: CLS; LOCATE; PRINT

Istruzione WAIT

Descrizione

Sospende l'esecuzione del programma fino a quando il modello di bit specificato non viene letto da un porta di input.

Sintassi

WAIT *numero_porta*, *AND_espressione*[,*XOR_espressione*]

Note

- *numero_porta* corrisponde a un'espressione intera (da 0 a 255) che identifica la porta.
- *AND_espressione* corrisponde a un'espressione intera che WAIT confronta con il valore della porta utilizzando l'operatore AND.
- *XOR-espressione* corrisponde a un'espressione intera che WAIT confronta con il valore della porta utilizzando l'operatore XOR
- I dati provenienti dalla porta specificata vengono confrontati prima con *XOR_espressione* se questa è stata fornita. Il risultato viene quindi confrontato con *AND_espressione*. Se il risultato è zero, WAIT continua a leggere i valori della porta; altrimenti il QBasic esegue l'istruzione successiva.

Esempio

```
WAIT 45, 64
```

Istruzione WHILE/WEND

Descrizione

Restituisce una serie di istruzioni fino a quando la condizione specificata viene soddisfatta.

Sintassi

WHILE *condizione*
[*istruzione*]
WEND

Note

- *condizione* corrisponde a un'espressione booleana. Fino a quando la condizione rimane soddisfatta, il QBasic esegue le istruzioni all'interno del loop.
- *istruzione* corrisponde a un elenco qualsiasi di istruzioni.
- WHILE/WEND è una vecchia costruzione per la creazione di un ciclo. Si preferisce l'utilizzo di DO.

Esempio

```
i = 0
WHILE i < 100
    PRINT i
    i = i + 1
WEND
```

Istruzioni correlate: DO UNTIL; DO WHILE; EXIT

Istruzione WIDTH

Descrizione

Imposta il numero di colonne sul video o su un'altra periferica o imposta la lunghezza di un file.

Sintassi

WIDTH [*colonne*][*,righe*]
oppure
WIDTH #*numero_file*, *colonne*
oppure
WIDTH *nome_periferica*, *colonne*
oppure
WIDTH LPRINT *colonne*

Note

- *colonne* corrisponde al numero di colonne. Il valore deve essere 80 o 40 per lo schermo.
- *righe* corrisponde al numero di righe di testo che appaiono a video. Il valore può essere 25, 30, 43, 50 o 60 a seconda dell'adattatore grafico di cui si dispone e della modalità video.
- *numero_file* corrisponde al numero di file assegnato al file nella relativa istruzione OPEN.
- *nome_periferica* corrisponde a una stringa contenente il nome della periferica desiderata.
- La parola chiave LPRINT imposta il numero di colonne per la stampante.
- Dopo aver specificato la larghezza per una periferica, le istruzioni output sovrappongono automaticamente la riga successiva quando la larghezza viene superata.

Esempio

```
'Monitor EGA
WIDTH 80, 43
FOR i = 1 TO 100
   PRINT i
NEXT i
```

Istruzioni correlate: LPRINT; LPRINT USING; PRINT; PRINT USING; SCREEN

Istruzione WINDOW

Descrizione

Definisce le coordinate logiche per la finestra corrente.

Sintassi

WINDOW [[**SCREEN**](*x1, y1*)-(*x2, y2*)]

Note

- La parola chiave SCREEN inverte il sistema di coordinate in modo che i valori *y* aumentino via via che proseguono dall'alto verso il basso dello schermo.
- Le coordinate *x1*, *y1* e *x2*, *y2* specificano le coordinate logiche della finestra.

Esempio

```
SCREEN1
WINDOW (0, 0)-(50, 50)
LINE (10, 10) -(40, 40), 2, B
```

Istruzioni correlate: CLS; SCREEN; VIEW

Istruzione WRITE

Descrizione

Scrive i dati sul video o su un file sequenziale.

Sintassi

WRITE[[#]*numero_file*,]*elenco_espressione*

Note

- *numero_file* corrisponde al numero assegnato al file nella relativa istruzione OPEN. Se si omette *numero_file*, il QBasic scrive i dati a video.
- *elenco_espressione* corrisponde a un elenco di una o più variabili o espressioni separate tra loro da virgole.
- WRITE posiziona una virgola tra le espressione all'interno del file, operazione che non viene eseguita da PRINT.

Esempio

```
OPEN "TEST.DAT" FOR OUTPUT AS #1
WRITE #1, "TEST", 5, 3.21, "END"
CLOSE #1
```

Se si lancia questo programma, viene visualizzato quanto segue:

```
"TEST",5,3.21,"END"
```

Istruzioni correlate: PRINT

APPENDICE A

Parole chiave non gestite da QBasic

QuickBasic

ALIAS
BYVAL
CDEL
COMMAND$
EVENT
$INCLUDE
Int86
Int86x
Interrupt
InterruptX
LOCAL
SADD
SETMEM
SIGNAL
UEVENT

GW-BASIC/BASICA

AUTO
CONT
DEF USR
DELETE
EDIT
LIST
LLIST
LOAD
MERGE
MOTOR
NEW
RENUM
SAVE
USR

APPENDICE B

Indice funzioni e istruzioni

ABS (funzione)
ASC (funzione)
ATN (funzione)

BEEP (istruzione)
BLOAD (istruzione)
BSAVE (istruzione)

CALL (istruzione)
CALL ABSOLUTE (istruzione)
CDBL (funzione)
CHAIN (istruzione)
CHDIR (istruzione)
CHR$ (funzione)
CINT (funzione)
CIRCLE (istruzione)
CLEAR (istruzione)
CLNG (funzione)
CLOSE (istruzione)
CLS (istruzione)
COLOR (istruzione)
COM (istruzione)
COMMON (istruzione)
CONST (istruzione)
COS (funzione)
CSNG (funzione)
CSRLIN (funzione)
CVD (funzione)
CVDMBF (funzione)
CVI (funzione)
CVL (funzione)
CVS (funzione)
CVSMBF (funzione)

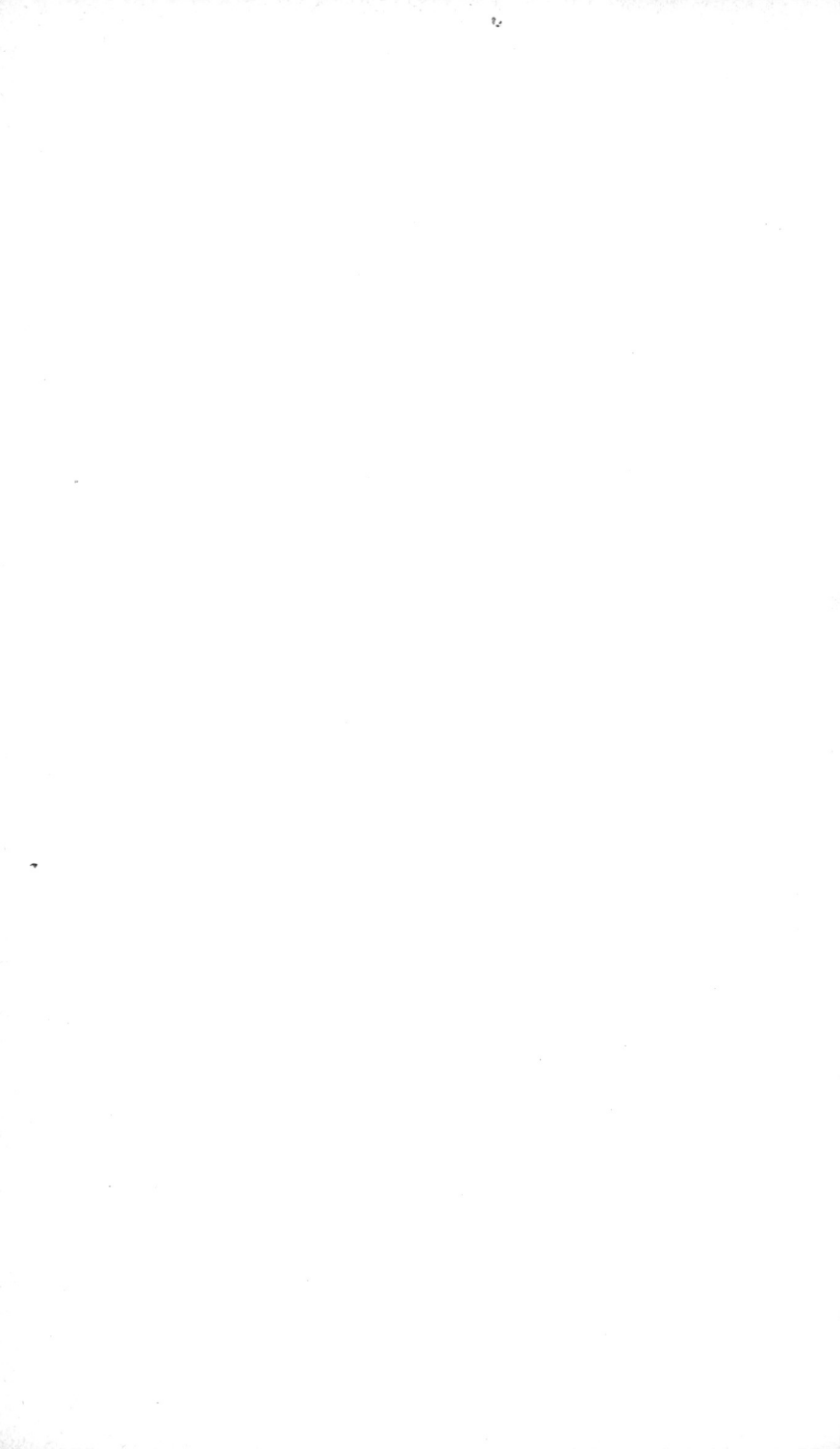
DATA (istruzione)
DATE$ (funzione)
DATE$ (istruzione)
DECLARE (istruzione)
DEF FN (istruzione)
DEF SEG (istruzione)
DEFDBL (istruzione)
DEFINT (istruzione)
DEFLNG (istruzione)
DEFSNG (istruzione)
DEFSTR (istruzione)
DIM (istruzione)
DO UNTIL (istruzione)
DO WHILE (istruzione)
DRAW (istruzione)

END (istruzione)
ENVIRON (istruzione)
ENVIRON$ (funzione)
EOF (funzione)
ERASE (istruzione)
ERDEV (funzione)
ERDEV$ (funzione)
ERL (funzione)
ERR (funzione)
ERROR (istruzione)
EXIT (istruzioni)
EXP (funzione)

FIELD (istruzione)
FILEATTR (funzione)
FILES (istruzione)
FIX (funzione)
FOR (istruzione)

APPENDICE C

Codici scan

Tasto	Codice	Tasto	Codice	Tasto	Codice
Esc	1	A	30	F1	59
! o 1	2	S	31	F2	60
@ o 2	3	D	32	F3	61
# o 3	4	F	33	F4	62
$ o 4	5	G	34	F5	63
% o 5	6	H	35	F6	64
^ o 6	7	J	36	F7	65
& o 7	8	K	37	F8	66
* o 8	9	L	38	F9	67
(o 9	10	: o ,	39	F10	68
) o 0	11	" o '	40	F11	133
_ o -	12	~ o `	41	F12	134
+ o =	13	Shift sinistra	42	Bloc Num	69
Backspace	14	| o \	43	Blocc Scorr	70
Tab	15	Z	44	Home o 7	71
Q	16	X	45	↑ o 8	72
W	17	C	46	PAG↑ o 9	73
E	18	V	47	–grigio	74
R	19	B	48	← o 4	75
T	20	N	49	Centro o 5	76
Y	21	M	50	→ o 6	77
U	22	< o ,	51	+grigio	78
I	23	> o .	52	Fine o 1	79
O	24	? o /	53	↓ o 2	80
P	25	Shift destra	54	PAG↓ o 3	81
{ o [	26	PrtSc o *	55	Ins o 0	82
} o]	27	Alt	56	Canc o .	83
Invio	28	Barra spazio	57		
Ctrl	29	Caps Lock	58		

GUIDA RAPIDA

MS-DOS® QBasic™

Kris Jamsa propone con questa guida un manuale di rapida consultazione per QBasic, il linguaggio di programmazione BASIC che accompagna il sistema operativo MS-DOS 5. Attraverso illustrazioni esemplificative, la guida introduce il lettore all'ambiente MS-DOS QBasic: tipi di dati, variabili e operatori, differenze con GW-BASIC e BASICA. Ideale per reperire informazioni chiare e sintetiche sull'avvio del sistema, l'uso dei menu, dei sottoprogrammi e delle funzioni di QBasic, il volume costituisce una guida chiara e di facile consultazione.

ISBN 88-386-0247-6

Lire 20.000 (i. i.)